PROGRAMMING LEARNING GAMES
Identification of game design patterns in programming learning games

Master Degree Project in Informatics
One year Level 22.5 ECTS
Spring term 2019

Ander Areizaga

Supervisor: Henrik Engström
Examiner: Mikael Johannesson
Abstract

There is a high demand for program developers, but the dropouts from computer science courses are also high and course enrolments keep decreasing. In order to overcome that situation, several studies have found serious games as good tools for education in programming learning. As an outcome from such research, several game solutions for programming learning have appeared, each of them using a different approach. Some of these games are only used in the research field where others are published in commercial stores. The problem with commercial games is that they do not offer a clear map of the different programming concepts. This dissertation addresses this problem and analyses which fundamental programming concepts that are represented in commercial games for programming learning. The study also identifies game design patterns used to represent these concepts.

The result of this study shows topics that are represented more commonly in commercial games and what game design patterns are used for that. This thesis identifies a set of game design patterns in the 20 commercial games that were analysed. A description as well as some examples of the games where it is found is included for each of these patterns.

As a conclusion, this research shows that from the list of the determined fundamental programming topics only a few of them are greatly represented in commercial games where the others have nearly no representation. This shows potential spots for future research in games for programming teaching.

**Keywords:** Programming learning, game design patterns, serious games
# Table of Contents

1 Introduction ........................................................................................................... 1

2 Background ........................................................................................................... 2
   2.1 Programming in software development .......................................................... 2
       2.1.1 Programming concepts ........................................................................ 3
       2.1.2 Learning programming ........................................................................ 4
   2.2 Serious games for programming teaching ...................................................... 5
       2.2.1 Programming game examples ............................................................... 6
   2.3 Game design .................................................................................................... 7
       2.3.1 Game design patterns ........................................................................... 8

3 Problem ................................................................................................................ 10
   3.1 Method ............................................................................................................. 11
       3.1.1 Game selection phases ......................................................................... 12
       3.1.2 Initial scanning phase .......................................................................... 13
       3.1.3 Review of the games phase .................................................................. 14
       3.1.4 Game classification phase .................................................................... 15
       3.1.5 Pattern reviewing phase ....................................................................... 15
   3.2 Ethical consideration ...................................................................................... 15

4 Results .................................................................................................................. 17
   4.1 List of selected games ..................................................................................... 17
       4.1.1 Steam store games ............................................................................... 18
       4.1.2 Google Play store games ...................................................................... 18
   4.2 Game analysis .................................................................................................. 18
   4.3 Game classification ....................................................................................... 18
       4.3.1 Classification of all the games ............................................................... 19
       4.3.2 Steam store games .............................................................................. 20
       4.3.3 Google Play store games ..................................................................... 20
   4.4 List of game design patterns .......................................................................... 22
       4.4.1 Name: Block-based programming ....................................................... 22
       4.4.2 Name: Step by step program execution ............................................. 23
       4.4.3 Name: Visual Input and output representation ..................................... 23
       4.4.4 Name: Instruction-based programming ............................................. 24
       4.4.5 Name: Registers' representation ......................................................... 24
       4.4.6 Name: Commentary writing ................................................................. 25
       4.4.7 Name: Hidden game programming challenges .................................. 25
       4.4.8 Name: Program examples levels ......................................................... 26
       4.4.9 Name: Device hacking ......................................................................... 26
       4.4.10 Name: Program output log ................................................................. 26
       4.4.11 Name: Resource limitation ................................................................. 27
   4.5 General outcomes ............................................................................................ 27

5 Conclusions .......................................................................................................... 29
   5.1 Summary .......................................................................................................... 29
   5.2 Discussion ......................................................................................................... 30
   5.3 Future Work ..................................................................................................... 31

References ................................................................................................................ 33
1 Introduction

Learning programming has always been hard for novices, leading sometimes to a point of needing to dropout. Vhaldick, Mendes and Marcelino (2014, p.1) explain that there is an existing problem with a high global demand for software developers, which is not met as the number of computer course’s enrolments is decreasing and the failure and dropouts’ rates are high. This made researchers start to think in alternative ways of teaching programming, breaking the entry barriers that coding provokes in different development environments.

As Arnab et al. (2012, p. 159) state “in recent years considerable interest has been devoted to the pursuit of learning through, and with, digital games and particularly so-called Serious Games”. The increasing level of popularity, in the use of games for learning purposes, led to the development of several game solutions with the aim to address the existing problem with novices regarding learning programming. Xinogalos and Galgouranas, (2018) present a serious game solution for teaching programming with Java. As in other solutions, the results in this case show that most students prefer the game over the other teaching alternatives. Therefore, as stated in a previous Xinogalos paper (2015) the problem befalls on students not understanding correctly the main concepts in programming. This is eased with the smooth introduction of these concepts through the game, which proved to be more successful than other methods.

Nowadays a lot of game solutions are available to the public with a clear purpose of teaching programming. Vhaldic et al. (2014) and Miljanovic and Bradbury (2018) review several game solutions for teaching programming. As an additional note to consider, some of the solutions they review come from different research papers, where others come from commercial stores like “Steam” (Valve corporation, 2019) or “Google Play” (Google, 2019). In order to find what topics related to programming are being taught with these games, they analyse them against the ACM Computer Science Curricula from 2013. Within that document, there is an important section containing the different software fundamentals necessary in every computer science course appears. In this section, 4 main themes are identified with several topics that are tagged as fundamental. Those topics range from “concepts of algorithms” to “variables” and “conditionals” as well as different topics related to data structures among other relevant ones in the computer science field.

Within games there are several patterns that they have in common. The concept of game design pattern is developed by Björk and Holopainen (2003). They explain that as a common language while talking about game design is missing, they developed a framework “for the study of games based on game design patterns” (p.1). Game design patterns are defined as tools that can be used not only for problem-solving but also for supporting creative design works. These tools are presented in games that have something in common. In this case what they have in common is that the games share their main objective: programming learning.

Therefore, this dissertation focuses on the identification of the different game design patterns used in games for programming learning. For doing so, several games coming from commercial stores are selected for their further analysis, following the methodology described in the document. The special characteristic that this game design patterns should contain is the representation of the different topics relevant for the computer science field. The paper concludes with a discussion on the different computer science topics are found represented inside the games and which not, as well as future work.
2 Background

For the general understanding of this paper this chapter will serve as an introduction to several topics of interest. Therefore, this section has been divided into the next three main topics:

- Programming in software development
- Serious games for programming learning
- Game design

2.1 Programming in software development

For understanding the term programming, we must go back to the invention of the first machines and the concept of computing. Some of the first advanced machines that anticipated to some of the key designs of modern computers were those invented by Charles Babbage, Difference Engine and Analytic Engine. Even if any of these inventions were fully built, in Ada Lovelace’s collection of “Notes” her vision on the potential of these engines was similar to what today we know as computing (Fuegi and Francis, 2003, p. 16). Early definitions of computing only considered machines that executed different mathematical operations, starting from the abacus, but nowadays, computing consists in every machine that can manage different types of information for various purposes (Blackwell, 2002, p. 1).

Some of these machines have advanced until what we nowadays call computers, “programmable electronic devices that can process, store and retrieve data” (O’Regan, 2012, p. 23-24). These computers are formed by two main parts: hardware and software. Hardware is the physical part of the machine where software is the intangible part. Software is the programable part made by a programmer or a team of programmers (O’Regan, 2012, p. 33) and therefore here is where the concept of programming appears.

Several papers have appeared through the history trying to give an answer to “What is programming?”. Although the word programming has several meanings by itself, where one of them is to write programs in order fulfill a need or automate something, that will in some way pay off (Blackwell, 2002, p. 7). However, there have been several contradictions while defining programming through the history. There exist people that differentiate the words “coding” and “programming” by stating that “Programming is the process of drawing up the schedule of the sequence of individual operations required to carry out the calculation” (Hartree 1950, p. 111). But in the other hand, McCracken (McCracken, 1957) defines programming as “a process of translating from the language convenient to human beings to the language convenient to the computer” including the earlier explained coding.

So when we strictly talk about programming for software development we talk about a task of understanding a certain existing problem, that will follow-up with the process of designing a suitable solution that could solve the problem and coding the designed solution, which involves three main competences: comprehension, writing and debugging (Vhaldick et al., 2014, p.1), and maintaining it (Hoc, Green, Samurcay and Gilmore 1990, p.46).
2.1.1 Programming concepts
For understanding programming in general some of the concepts that are generally used inside the environment need to be explained. For this part, the Software Development Fundamentals of the Computer Science Curricula 2013 section is used as a source for identifying the different programming concepts of interest.

Inside the curricula, the Software Development Fundamentals are divided into 4 main units: algorithms and design, fundamental programming concepts, fundamental data structures and development methods. From this main units several topics related to them are identified summarizing different programming concepts of interest for this paper.

**Table 1** Software development fundamentals (ACM Computer Science Curricula, 2013)

<table>
<thead>
<tr>
<th>Units</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms and design</td>
<td>• Concept and properties of algorithms</td>
</tr>
<tr>
<td></td>
<td>• Role of algorithms in the problem-solving process</td>
</tr>
<tr>
<td></td>
<td>• Problem solving strategies</td>
</tr>
<tr>
<td></td>
<td>• Fundamental design concepts and principles</td>
</tr>
<tr>
<td>Fundamental programming concepts</td>
<td>• Basic syntax and semantics</td>
</tr>
<tr>
<td></td>
<td>• Variables and primitive data types</td>
</tr>
<tr>
<td></td>
<td>• Expressions and assignments</td>
</tr>
<tr>
<td></td>
<td>• Simple I/O including file I/O</td>
</tr>
<tr>
<td></td>
<td>• Conditional and iterative control structures</td>
</tr>
<tr>
<td></td>
<td>• Functions and parameter passing</td>
</tr>
<tr>
<td></td>
<td>• Recursion</td>
</tr>
<tr>
<td>Fundamental data structures</td>
<td>• Arrays</td>
</tr>
<tr>
<td></td>
<td>• Records/structs</td>
</tr>
<tr>
<td></td>
<td>• Strings and string processing</td>
</tr>
<tr>
<td></td>
<td>• Abstract data types</td>
</tr>
<tr>
<td></td>
<td>• References and aliasing</td>
</tr>
<tr>
<td></td>
<td>• Linked lists</td>
</tr>
<tr>
<td></td>
<td>• Strategies for choosing the appropriate data structure</td>
</tr>
<tr>
<td>Development methods</td>
<td>• Program comprehension</td>
</tr>
<tr>
<td></td>
<td>• Program correctness</td>
</tr>
<tr>
<td></td>
<td>• Simple refactoring</td>
</tr>
<tr>
<td></td>
<td>• Modern programming environments</td>
</tr>
<tr>
<td></td>
<td>• Debugging strategies</td>
</tr>
<tr>
<td></td>
<td>• Documentation and program style</td>
</tr>
</tbody>
</table>

Within these concepts it is of interest mentioning the concept of programming language. A programming language is what in the ACM Computer Science Curricula (2013, p. 155) is
described as a “medium through which programmers precisely describe concepts, formulate algorithms, and reason about solutions”. This means that the programmers make use of them for writing the different programs that the computers will interpret for running them. However, these programming languages are classified into various programming paradigms. There exist several programming paradigms, but the ones of interest to discuss in this dissertation, due to their more common usage for learning programming (Pears et al. 2007), are procedural programming and object-oriented programming paradigms. As an example of procedural programming languages Pascal and C are the most commonly used ones, where for object-oriented programming languages Java, C++ and the recent Python are the most common ones (Robin, Rountree and Rountree, 2003). The main differentiation between these two programming paradigms is that OO programming languages allow objects development inside the program. The other characteristics are similar between them, where programmers are provided with a specific syntax and semantic for making the different programs.

These programming languages are normally used around certain programming environments. These environments act as helping tools for programmers that let them carry out their tasks. There exist two main types of programming environments, the basic environment, where in Java, it consists of a simple text editor for editing Java files and some command line tools for compiling and executing programs, and more advanced Integrated Development Environments (IDE) that provide more features such as organizing programs and other support tools (Pears et al. 2007).

2.1.2 Learning programming

Despite programming being a “very useful skill” and a “rewarding career” (Robin, Rountree and Rountree, 2003, p. 137) it “has traditionally been considered a difficult topic, and research has shown that novices face several problems when learning to program” (Mannila, Pletomäki and Salakoski, 2006, p. 211). This is one of the main reasons for which several researches have tried to find an ideal learning process for programming.

Vhaldicks et al (2014) identify three essential competencies for the learning process to be complete: comprehension, writing and debugging. These competencies are transversal, and they are based on a set of skills, which means that the different skills may overlap among them and it will not be possible to acquire a whole competence at a time.

Comprehension is a competence based on understanding how the different programs work within the computer. One of the biggest weakness of the novices is not understanding how a computer interprets a program, which is based on reading the code line by line and understanding only the different commands that are explicitly coded (Vhaldick et al., 2014).

For understanding how to structure a program or how different parts of the code interact with each other, the students should be taught how to make different graphical models and representations of the programs. This helps in visualizing the program they want to make and makes it easier to understand the relation between the code parts. One of the things that difference experts from novices in programming is that while experts have efficiently organized and specialized knowledge schemas, novices are not used to them and therefore, they have difficulty understanding simple behaviors inside a program (Robin et al., 2003).

Rist summarizes what kind of advantages the expert programmer has as follows:
Expertise in programming should reduce variability in three ways: by defining the best way to approach the design task, by supplying a standard set of schemas to answer a question, and by constraining the choices about execution structure to the ‘best’ solutions.

Rist, 1995, p. 552

The development of the different schemas is something that takes part in the writing competence. In the writing competence, the first task that programmers need to make is selecting the programming language in which the program will be written. This will provide them with the different syntax and semantics that will be used, and the programming environment where the program will be executed (Vhaldick et al, 2014).

In different researches when it comes to the selection of the programming language there are different opinions on which one is the best for learning purposes. Pears et al. (2007) discuss which ones of the existing programming languages are more commonly used and what is the existing debate about the different suitability of the mentioned ones. The ones addressed as the most popular ones after the review of several papers are: Java, C and C++. These programming languages are not specifically designed for educational purpose, whereas Python, that is commonly debated as the solution to which they should move, has been designed for this purpose.

The writing competence ends up with a written program that now should be tested and debugged in order to search for the different errors that might have been made. It is important to note that the debugging and testing competence can take part many times during the writing of the program. The connection between debugging is stronger with the code comprehension than with the writing of it, as understanding the code and having better debugging skills will derive into a better writing of the code (Vhaldick et al, 2014).

These three competences take part in what is called the problem-solving process. This process consists in the different steps that a programmer will be taking for solving a concrete problem. These steps can be summarized into: the understanding of the problem, centered in the comprehension competence; determining how to solve a problem and translating this into the solution the computer can interpret, which makes use of the writing competence; and a last step requiring them to test and debug the program, where the debugging competence takes place (Winslow, 1996).

### 2.2 Serious games for programming teaching

Coelho, Kato, Xavier and Gonçalves (2011) explain that “serious computer games have emerged in recent history, but serious games have always been part of human culture as far as we know”. Of course, times have changed and what it was a serious game in ancient times differs a lot from the serious games being developed nowadays. The reasons for these changes might be, in some part, due to the advancements in technology that have occurred, where other facts could have taken part in it. Nevertheless, the concept of serious game has remained the same, where they consist of games with the aim of not only entertaining but also serve for a more serious purpose like teaching, simulating, health treatments etc.

Therefore, the serious game definition that will be used inside this paper is the following one:
Serious game: a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives.  

Zyda (2005, p.26)

As the serious game definition states, there exists several ambits where these kinds of games exist. One of these ambits is education, where education also contains several themes of interest within itself, as well as different ways of teaching with the use of games. One of the themes included within serious games for education is programming learning.

Muratet, Torguet, Jessel and Viallet (2009) explain that one of the reasons for which computer science courses create distaste is the lack of meaning and relevance certain assignments have. Therefore, in this direction they identified 3 approaches, where one of them is video games. For learning programming with video games two approaches have been used: creating and playing games, also explained by Vhaldick et al. (2014). The first one will consist in making or implementing games in diverse ways. These games should be short as the students need to complete them in a short amount of time.

The second approach will consist in letting the students learn by playing games. With this, the main idea is to motivate students to learn by giving them interesting activities to enhance with. They will also get shorter times between the theoretical part of learning and the practice of what they learned (Vahldick et al., 2014). From these two approaches, the one that fits into the serious game definition game mentioned before is the second one, where the user will play games to learn.

2.2.1 Programming game examples

There exists a variety of games about programming teaching throughout the gaming market. As an example of the games quantity available there exist two papers that focused on reviewing the available games. These papers are “A review of serious games for programming” by Miljanovic and Bradbury (2018) and “A review of games designed to improve introductory computer programming competencies” by Vahldick et al. (2014). The games shown in these papers are solutions that come mostly from research papers, but some of them come from commercial stores such as AppStore and Google Play or other kind of stores.

Some of the examples that appear in these papers are Mays Journey (Jenmali, 2016), RoboBug (Miljanovic and Bradbury, 2017) and Prog&Play (Muratet, Torguet, Viallet and Jessel, 2011). These 3 games come from different research papers as solutions for programming learning with different approaches.

Mays Journey is a game aimed for 12 to 18-year-old girls with the purpose of attracting them to the Computer Science field. In this game the players play as May who you will have to help in finding her friend and repairing the broken world. For doing so, the players will have access to a user interface. With this interface they will have to program a certain solution that will make the objects in the maze execute specific actions, letting May walk through the map. The program is coded in a language made specifically for the game (Jenmali, 2016).
RoboBug is a serious game intended for students that are learning to debug programs. It is a puzzle-type game where the puzzles that appear inside the game will require the players to search for different bugs in the programs and solve them (Miljanovic and Bradbury, 2017).

Prog&Play is a serious game dedicated to programming practice. It is based on a real-time strategy game called Kernel Panic. Based on this game the developers created a scenario for the start of a campaign, which they identified as a solution for mapping learning objectives into the game. In this campaign the players will have to change the code for moving or doing certain actions with the bits and complete the different tasks (Muratet et al., 2011).

All these 3 games have a different approach to achieve the goal of teaching programming. The problem that appears when evaluating these games, is that, as they come from research papers, sometimes the authors do not offer the solution for the readers to play them. As an example, Miljanovic and Bradbury (2018) explain that the game Mays Journey game was not available to download during their evaluation process. However, Prog&Play and RoboBug were accessible for them to play and evaluate.

Therefore, as explained by Miljanovic and Bradbury (2018) and Vahldick et al. (2014) the game availability, when they belong to research papers, is a problem. It is common that in order to get these game solutions for their further analysis, researchers are needed to contact the authors of the different papers where the games come from. Along with this, they also need to get their permission to use them, where sometimes it is possible that they will not provide nor the game nor the permission in any ways.

Nevertheless, with commercial games like Human resource machine analyzed by Miljanovic and Bradbury (2018) the availability was not a problem as it is available in the different commercial stores for its purchase. This game is based in a puzzle game where the players are provided with several levels. In each of these levels the players will have to make the program by dragging and dropping blocks that represent certain actions. When pressed the button play, the program will run, and the workers will behave as told by the program. The objective in each of the level’s variates, making the players to complete certain tasks in each of them.

### 2.3 Game design

Before getting into a more complex explanation of the design in games, first a clear definition of what a game is should be made. In this case, the following no rigorous definition, as stated in the book, will be used as a game definition:

*A game is a type of play activity, conducted in the context of a pretended reality in which the participant or participants try to achieve at least one arbitrary, nontrivial goal by acting in accordance with rules.*

Adams, (2014, p. 3)

Adams (2014) identifies four essential elements in a game. These elements consist of playing, pretending, a goal and rules.

Play consists in entertainment where the user must interact with the source. As games are formed by the need of humans to entertain themselves, playing is one of the essential elements of them. There are certain similarities with playing and other forms of
entertainment like reading books or watching films, but what makes playing as a unique form of entertainment is the freedom and the different experience you get every time you play. (Adams, 2014)

Pretending consist, as the words definition says, in behave as if the things that are happening were real. In a game, a player could pretend things that in the real world would be impossible to happen. In the same way that some films create fictional worlds where things that happen or appear inside them have no explanation outside that context, in games the players are presented with a similar situation with the exception that in this case they take part in the action (Adams, 2014).

A goal is something that all the games must have for giving the players a certain objective inside the game. There exist games that have only a goal but there also exist games that could contain various goals in it. The goal of a game is directly defined by its rules and it depends on the designers as they can define it any way they like (Adams, 2014). As an example, in a game like Pacman the goal is as simple as winning the current game you are playing by eating all the dots.

Finally, the rules consist everything you can and cannot do within a game. Sometimes these rules are not specifically written, but you know they exist inside the game just by the experience you get by playing it. There are several things that the rules define in a game, like the meaning or relationships between certain symbols inside the game, the challenges and actions the game offers the player, activities progression that makes up the game, the objective of the game (the goal), the condition that ends the game if it has one, and the rules about the rules or in other words the rules that explain when are the rules applied of it they can change through the game (Adams, 2014).

However, Hunicke LeBlanc and Zubek (2004, p.2) explain that games can be broken into 3 different components: Rules, System and “Fun”. These components have their design counterparts as: Mechanics, Dynamics and Aesthetics forming the framework MDA. Mechanics, in this case, describe the different components of the game that represent the actions, behaviors and control mechanism given to the player inside the game. Dynamics are the different interaction the mechanics and the aesthetics have in the game. As an example, using the mechanic of the jump when running for longer jumps. Aesthetics is the looks of the game that provoke certain emotions in the players. All these components all together form what is considered the design of the game, or in other words, what makes a game unique from the others.

2.3.1 Game design patterns

Every game is different in some way, it can be due to its form of approaching different themes, innovative mechanics that no game has tried before or maybe its music is the one thing that stands out. Therefore, there is a need of a common language when talking about the design of the games, or when you are trying to analyze the game play so everyone can understand the same thing without requiring much effort (Björk, Lundgren and Holopainen, 2003, p. 181)

Björk, Lundgren and Holopainen (2003, p.185) define game design patterns as tools to support creative design work, where in other design environments, design patterns are defined as problem solving solutions. This is due to the increase in creativity needed to design a game comparing to other design environments.
For identifying a game pattern, they developed a structural framework. This framework consists of the following sections: name, description, consequences, using the pattern and relations (Björk, Lundgren and Holopainen, 2003, p. 185).

- The name should be short, specific and idiomatic.
- The description should explain how the pattern affects the structural framework and should be concise with the inclusion of notes with games it was found on.
- The consequences should explain the different trade-offs and consequences of using the pattern.
- Using the pattern should give a general information on how to apply the pattern in a game.
- The relations section should explain what relations exist between different game design patterns that have already been stated.

Inside the same document, one example of design patterns they found in the game paper rock scissors is explained. This pattern is based on the relation the different options have inside the game, that forms the shape of a triangle. The rock wins against the scissors, the scissors win against the paper and the paper wins against the rock. As an example of another game that uses the same mechanic, there exists Quake, where the relation between the weapons and the monsters is like the one explained in the rock papers scissors.

In collaboration with other researchers, Björk has also addressed several game design patterns that belong to other areas and topics. In the area of dark patterns, Björk, Zagal and Lewis (2013) identify several patterns that are considered “bad designs”. These patterns as defined by them, are “patterns used intentionally by a game creator to cause negative experiences for players that are against their best interests and happen without their consent” (p.3). With this description of what is a dark pattern, they provide with several examples divided in categories that affect the players in several ways. One of the categories is named “Temporal Dark patterns” which are based on making the players waste their time in pointless tasks. One of the examples they give is “Grinding”, which consists in making the players do repetitive and tedious tasks in order to obtain certain objects or to continue with the game. One of the games used as an example for this pattern is the famous massive multiplayer online game World of Warcraft. In this game the players are asked in several times to kill the same enemy a specified amount of times, making the tasks repetitive and time-wasting.

In order to identify games from other topics too, Björk and Brusk (2009) identify several patterns from games that belong to dialogues in games. Dialogues in games have several functions. Some of these functions are to give and coordinate information between the different characters available in the games, for creating a social cohesion or for making the players and the characters to understand each other. The design patterns that are identified in this document about dialogues in games, consist in the different ways the dialogues are represented within the games that are examined in the papers and the interaction of this with other mechanics in the games. As an example, they put the gameplay integrated conversations, where dialogues can take place at the same time as other actions leaving the player in a situation where they have to convince the guards to let them go through while they are being chased by monsters.
3 Problem

As explained in the introduction and section 2 of this dissertation, there is an existing problem in computer science courses. While there is a rising demand of software engineers required, the number of computer science course’s enrolments are decreasing, and the rate of dropouts remains high. This in conjunction with the entry barriers that programming has it makes a huge problem for novices to finish their studies. Therefore, different researchers state that for solving this problem, making serious games for programming teaching is a good approach.

With this motivation several games for programming teaching started appearing with different approaches. Examples like Mays Journey (Jenmali, 2016), RoboBug (Miljanovic and Bradbury, 2017) and Prog&Play (Muratet, Torguet, Viallet and Jessel, 2011) presented in the background or “A review of serious games for programming” by Miljanovic and Bradbury (2018) and “A review of games designed to improve introductory computer programming competencies” by Vahldick et al. (2014) which analyze several games, show that there has been an increase in the research of these types of games. These games are not only being developed in the research environment as commercial stores also provide several solutions. Looking at the great number of games available to the general public in the stores, it was of interest to analyze how do they approach programming in each of them.

The problem with the games available in commercial stores is that they don’t offer a clear mapping of the different programming concepts. Nevertheless, as some of these games focus in representing the same programming concepts inside them, they can be classified and analysed regarding this information. Besides, several game design patterns (Björk, Lundgren and Holopainen, 2003) are used in the games to represent the same programming concepts in a similar way.

Therefore, the main goal of this study is to analyze several of the existing commercial games that focus in programming learning and explore which topics (of the ones explained in the programming concepts background) are covered with them. The purpose of this analysis is to find out what game design patterns, that cover similar topics, are used in the games and see if there exists a certain similarity within them. This will also lead to information about what topics are with fewer games or none, in order to find potential spots for future researches. From this information, it can be deduced whether current state of games for programming learning is good enough to consider them as good tools for education.

Along with this, these commercial games have shown to have similar ways of representing the different topics mentioned before. As an argument for supporting this statement, in this study several game design patterns used in these games for representing the most represented topics are identified and explained. The purpose of this list of game design patterns is for their use in future development as tools for creative inspiration. It also shows what has been done in the field and what is still missing or is possible to do in the future.

Specifying the different objectives of these study into a singular research question is, “which of the fundamental programming concepts are represented in commercial games for programming learning and what game design patterns are used for representing them?”.
3.1 Method

For developing this method, the searching criteria was based in the ones used in “A review of serious games for programming” by Miljanovic and Bradbury (2018) and “A review of games designed to improve introductory computer programming competencies” by Vahldick et al. (2014). However, in these papers the searches were mostly done in databases searching for research papers. But, as in this study the games come from commercial stores, the searching criteria needed to be adjusted to new parameters.

The method that is used for doing this study consists in a systematic analysis of games for learning programming. For doing so, several phases are defined for this process. These phases are focused in the searching and selection of the games that will be used in the study, review of the selected games for their classification and a final phase for identifying and reviewing the different patterns.

![Game selection phases](image)

**Figure 1** Flowchart showing the different phases that the method will follow

The first phase is the searching phase. This phase consists on researching the existing game solutions for teaching programming. In order to find the solutions that were going to be analyzed in this paper a certain selection criterion was defined, depending on the source and platform the games were coming from. These selection criteria are defined in section 3.1.1. The distinction between sources and platforms is made due to the different search engines used in the stores where games of this kind can be obtained and the availability of games inside them.

The following phases will be the sorting phase, the initial scanning phase and the selection of the games phase.

- In the sorting phase, the different games that appear in the searches are sorted by some defined parameters. These parameters are explained in section 3.1.1, where a more concrete explanation of the phase is made.
- In the initial scanning phase, a brief analysis of the current first 30 games that appear in the each of the sorted lists is made. The number of chosen games is 30, due to the low popularity of the games that appear after that point and the lack of any special
characteristic that made them unique from the others. This phase is explained in detail in section 3.1.2. The last phase is the selection of the games.

- In this phase the games that have been previously analyzed will be sorted by their focus in programming, and the first 10 games that are more programming focused are selected.

These phases, in conjunction with the search phase, consist in the selection of the games following a specified selection criterion. These games are the ones that are used in this study in the following phases.

After the game selection is done, the reviewing phase takes place. In this phase, the games that have been selected are systematically analyzed and reviewed. The purpose of the analysis is to gather important information about the design and style of the games that have been selected. This information will be of use in the study to understand the different objectives of the games and what programming concepts they cover with the games. The process of this analysis is explained in section 3.1.3.

Once the analysis is done, the classification phase takes place. In this phase a classification of the games regarding to the topics they cover is made. These topics are extracted from the ACM Computer Science Curricula 2013, whereas explained in section 2.1.2 in the background, they provide a list of software development fundamental topics.

With this classification of the games done, the topics that contain a mayor number of games that focus on them are highlighted. Its purpose is to identify the different game patterns used in the games that focus in those topics. For the explanation of the identified game patterns, the framework explained in section 2.3.1 is used. Along with these topics, the ones that have less focus in the games are highlighted as they represent potential spots for future researches in serious games for programming learning.

3.1.1 Game selection phases

For the different phases that focus in the selection of the games, a certain protocol was followed. This protocol consists on the following steps:

1. List of the games for programming learning available in the different commercial stores, using a different search method in each of the commercial stores that is used.
2. First process of sorting the games for their selection. Different methods are used for different stores.
4. Selection of the final games that are used in the study.

The first list of games is acquired by doing a search in the different commercial stores that act as a source, where in this case “Steam” (Valve corporation, 2019) and “Google Play” (Google, 2019) are the ones that are used due to the number of users they have in their respective devices. Steam is the most popular store used for acquiring commercial games for computers, with peaks of 16 million users at the same time (Valve corporation, 2019), while Google Play store is the most popular one for downloading games in Android devices. The AppStore, made for apple devices, was discarded due to the lack of apple devices of the author and the similarity of games they contained while comparing it with Google Play store. The search of this games is made differently in the stores due to the different search engine they use inside the stores.
In Steam, the games are searched by selecting the tag programming and the tag game available inside the store. After this classification is made, the games are sorted by relevance. This shows up a list of games that the community decided contains programming content inside and are sorted by the relevance of the games inside the store. After this preselection of games is done, the second preselection is made by using a page called “steamspy.com” (Galyonkin, 2015). In this page, the games are sorted by the approximate number of owners that have these games. From this page the most popular games with the tag programming are preselected. This ends up with a list of 35 games due to the possibility of some of them being discarded in the following steps of this phase.

There exists one problem with this page and is that the approximate information is not accurate enough with games that are not so popular. Therefore, the order in which the games start appearing sometimes varies due to the inaccurate number it provides. To solve this problem another step in the sorting phase is made by using the page “playtracker.net/insight” (Marijan, 2019). In this page you can search for games and see a popularity rating as well as a more accurate number of owners of the different games. After this part of the selection phases, an approximate list of 30 games with programming content inside them is developed. The number of games selected is 30 as after the 30th game that appeared in the sorting process, the results were games that had very few users and did not have anything special comparing them with more popular games.

With the list of 30 games developed, they are submitted to the initial scanning of the games as explained in section 3.1.2. After this scanning of the games is completed, a selection process of the 10 games that are more programming focused is made. The outcome is the list of games that are being analyzed and reviewed.

In Google Play store, the games are searched by using the keywords “programming” and “games”. This search shows not only results containing games for programming learning but also several apps focused in programming learning. These apps are discarded due to the lack of game characteristics they should have had in order to be considered a serious game and therefore be accepted in this study. The games are sorted by the number of downloads they have, information available by visiting the store page of each of the games.

From this search, the first 30 games are listed for the initial scanning phase. This analysis is the same one that was done for the Steam games. With this analysis a final list of 10 games that focus more on programming from the Google Play store is made. These selected games are the ones that are analyzed and reviewed in the same way as the Steam store games.

The number of games that are going to be analyzed was established as 10 games for each store, 20 in total, as with this number a broad enough spectrum of programming learning games is represented. Also, each of these games was acquired by the author what implied a monetary disbursement.

3.1.2 Initial scanning phase
In the initial scanning phase, the list of 30 games coming from the sorting phase are used. As the objective of this phase is to acquire a first insight on what the games are about and how they focus in programming teaching, they are analyzed systematically. This analysis consists in using the description given by the developers and an explanation of the main objective of the game. The description is taken from the store pages of the games, where the developers put a brief description of what the game consists on. However, the main objective is
explained by the author of this paper with the information acquired while watching the videos available inside the store page or YouTube (Google, 2019) videos containing any kind of gameplay or explanation of the game. In general, this process takes about 30 minutes per game of gathering information about the game from the mentioned sources.

At the end, an evaluation of the general programming focus of the game is done with the information acquired in the previous steps of the protocol. This evaluation is classified into 5 levels: Low, Low-Medium, Medium, Medium-High and High. The information for these analyses is taken from the Steam page of the games and by watching “YouTube” videos of them in order to get a previous insight on what the programming concepts represented in the game are. This evaluation is based on the amount of programming content available in the games and the way its represented, for every time a programming concept appeared in one of the mentioned sources, a point was given to the game and the more points it had the higher the evaluation of the game was. For the making of the evaluation an explanation of the punctuation is given in each of the games. This explanation is developed by the author so it might be biased.

3.1.3 Review of the games phase

For the phase of reviewing the games, 10 games coming from Steam store and 10 games coming from Google Play store are used. Within this analysis the following 3 main questions are answered for each of the games:

- What does this game consist in? What is its purpose?
- How is the game designed?
  - User interface
  - Game mechanics
  - Characters
  - Representation of the player
- How are programming concepts represented in this game?

The objective of answering these 3 questions with this analysis is to identify the different game design patterns used in the games, specifically those who focus in representing programming concepts with them. In the review of the game these 3 questions are represented in 3 different sections. These sections are named as follows: description, design of the game and programming concepts represented inside the game.

For making this analysis, the games that have been selected will be acquired and played if possible. If the acquisition of certain games is not possible, due to device incompatibilities or lack of monetary resource, a thorough analysis of the different content available in the store and YouTube videos will be made as well as a research for pages describing the content of the game. In case that the second situation happens a notification in the analysis will appear to explain the specific situation. The amount of time that each of these games are played depends on the information given throughout the game. The game is going to be played until the author considers he has enough information about the game to complete the analysis. However, a look at more advanced levels is made to see if anything new of relevance is presented in them. This look is made by searching for Youtube videos that contain gameplay about them.
3.1.4 Game classification phase
For the game classification the topics that appear in the software development fundamentals section of the ACM Computer Science Curricula are used. The information gathered from the reviewing phase, in specific the section of programming concepts represented inside the game, is used as a source for this classification. A table for each of the main themes explained in the curricula is made. In these tables, the games are placed in the left side and the topics in the top side. If the game contains explanations or make use of the different concepts that each topic contains, a mark is placed in the square of the game and the topic it contains.

After these tables are done, the topics that contain more games and those who contain none or very few games are identified. These are the topics of interest for the study, as they represent “what has already been done and how” and “what is still missing to be done or could be done in future researches”. The games that cover these topics are the ones that go to the next phase, the pattern reviewing phase, in order to see how things are done inside these games.

3.1.5 Pattern reviewing phase
The pattern reviewing phase will consist on identifying and reviewing the different game patterns available in the games selected after the game classification phase. These games are the ones that contain representation of the topics that have more games that cover them. For identifying the game patterns, the information acquired in the review of the games phase is used.

After this identification is done, the framework explained in section 2.3.1 for developing game design patterns is used. The intention of using this framework is to state all the information regarding to that design pattern inside the same place, containing examples and explanations of their usage. The particularity that these game design patterns must contain is the representation of the specific topics selected in the classification of the games phase. All this means that for every pattern identified, the information regarding to their name, description, consequences, using the pattern and relations will be placed in a list for each selected topic. However, for this study the framework is slightly modified as it only has a description of the pattern, an explanation on how it is used and examples of games where it was found.

3.2 Ethical consideration
As explained in section 2.3.1, between design patterns there exist some denominated as “Dark Patterns”. These patterns are made by the creator of the games to intentionally cause negative experiences to the players without their consent. Some of these will waste the time of the players by making them perform repetitive tasks or even spending money for continue playing. Among this design patterns several main categories have been defined. These main categories are: Temporal Dark Patterns, Monetary Dark Patterns and Social Capital-Based Dark Pattern. The first one is based in wasting the players time, the second one is based on wasting the players money for going through the levels or acquiring better equipment and the last one is based on making social interactions obligatory if they want to advance in the game.
Apart from these dark patterns, there exist other that Björk, Zagal and Lewis (2013) have denominated as “shades of gray”. These game design patterns are ones that sometimes might cause negative experiences for players without their consent or work against their best interests. As examples, they divide them into 3 categories, Encouraging Anti-Social Behavior, Psychological Tricks and Games for Other purposes. The first category is based around making the players commit certain illegal activities or wrong behaviors, the second one is based on psychological tricks to make players want to play more and the last one is for making games with other purposes but without telling the players.

For the purpose of this dissertation, if any of these dark patterns are identified inside the games that are going to be analyzed within it, the danger and the various problems it can cause to the players will be addressed. This will give advice to those who want to use one of the identified game design patterns in case the one that they are interested on has disadvantages such as harming their players in an unethical manner and without their consent.
4 Results

In this chapter, the different results of the study are showed and explained. In the different sections inside this one show summarized results highlighting the important outcomes of the study. If a detailed information is needed for any of the sections, this is placed into the different appendices that the paper contains.

4.1 List of selected games

In this section the lists of the games selected from the commercial stores that have been used for the study is shown. These games are the ones that were submitted to the last analysis and classification to see the state of programming learning games in the commercial environment.

For the selection of the games the process mentioned in the method section was followed. In the Steam store a first search was made by using the different parameters that were mentioned. This search led to the first list of games that appears in appendix A. These are all the games that appear in the Steam store tagged as programming games by the community, which means that they could contain programming representations inside them.

The second list of games that appears in appendix A, is formed after sorting them with the page called “steamspy.com”. In this page, the games can not only be sorted by the mentioned programming tag available in Steam, but also by the number of owners they have. After the sorting, a list was made including the first 35 games with a greater number of owners. As the games that appear in the page with 20,000 owners or less are not sorted properly, the first 35 games were chosen in case some relevant solutions got discarded before going through the next steps of this phase.

As mentioned before, the results in this page show a lack of accuracy with the data of those ones that are not very popular. Therefore, the games that appeared in the second list got sorted by using another page. This page is “playtracker.net/insight” (Marijan, 2019) where the information about the number of owners is more accurate and a value of the current popularity of the games is available. Out of that page the first most popular 30 games were chosen. The number 30 was representative enough to proceed to the next phases.

In Google Play store, the search method was different as mentioned in section 3.1. The keywords used were “programming” and “games” and the results that appeared where games for programming learning and several apps that were discarded due to their lack of game content inside them. From this search, it was not possible to select the first 30 games as the amount of games that appeared was very low in comparison with Steam store, therefore, the first 15 games that appeared as the most popular were selected and placed in a list shown in appendix A.

After this pre-selection of the games was finished, the scanning games phase was done. This scanning phase is detailed in appendix B and its process is explained in section 3.1.2. From this scanning phase, the 10 games from each store that showed to be the most programming focus were selected and placed into the lists shown in this section. This process is the same for both stores. From this phase a total of 20 games, between both stores, were selected as this was considered representative enough of the current state of the games for programming learning.
4.1.1 Steam store games
1. 7 Billion humans
2. Else Heart.Break()
3. Human resource machine
4. Shenzhen I/O
5. While True: learn()
6. Marvellous Inc.
7. Exapunks
8. Robots: Create AI
9. Spacechem
10. TIS – 100

4.1.2 Google Play store games
1. Lightbot: Code Hour
2. Gladiabots
3. Coddy: World on algorithm Free
4. Algorithm City: Coding game for kids
5. Coding planets 2
6. Spritebox: code Hour
7. Coding game for kids – Learn to Code with Play
8. Human resource machine
9. Programming for kids – Learn coding
10. Coding planets

4.2 Game analysis
After the games have been selected, the analysis of the 20 games was done. This analysis is presented in appendix C, with all the games answers to the questions defined in section 3.1.3. For making this analysis, each of these games was acquired and played from 1h to 1h and a half, depending on how much time it took the author to get to the point where the main mechanics of the game were presented.

With this information, a general idea of what the game consisted on was acquired. The problem is that sometimes, new mechanics and concepts are presented later in the games. As playing the whole game would have required a lot of time, some Youtube videos containing gameplay of the more advanced levels of all the games were searched. In case some of these videos showed that new relevant mechanics or concepts related to programming were presented later in the games, they were considered and explained in the analysis using the mentioned videos as sources.

In general, each of this analysis took an approximate amount of 2 to 3 hours to write, where during the playing process, several notes for writing the analysis were taken.

4.3 Game classification
The selected games were classified according to the ACM Computer Science Curricula’s software development fundamentals after their analysis. These fundamentals are divided into 4 main areas, each of which has several different topics that according to this document every course that has the objective of teaching programming should include.
In appendix D, the tables showing the whole game classification are made but for this section a more simplified versions of the tables are included, only showing those topics that appear to be of interest during the process. This section is also divided between the games that come from the Steam store and those who come from the Google Play store. This division is made due to the different platforms they belong to and to identify if there are significant differences between them. For a topic to appear in each of the tables, at least half of the games or more from that store should have a representation of that topic inside the game.

4.3.1 Classification of all the games

After the game classification is done, the different tables that appear in appendix D were made. In these tables, the different topics were placed in the top side and the games were placed in the left side, where a mark in the field corresponding to the topic and the game it was represented in was placed. Within these tables, the information of the number of games that represented certain topics was acquired and therefore, a diagram containing the different number of games that represent the different topics was made as shown in the Figure 2.

![Diagram of the classification of the topics](image)

**Figure 2** Diagram of the classification of the topics
In the diagram shown in figure 2, the topic that was mostly represented inside the games with a number of 20 games is placed in the top side, where the ones that got the less representation in the games, that in this case is 0 games, is placed in the bottom side. From the results shown in this table, the topics that have less than the half of the games representing them are considered as topics that have very few representations and therefore, treated as future possible research questions.

4.3.2 Steam store games
From the information acquired after the analysis of the games that came from the Steam store, the following topics have been highlighted as those who contain more representation inside the games. This information is mainly acquired by the “programming concepts represented in the game” section made in each of the analysis of the games available in appendix C. In table 2, the topics, that had more representation in the games, are placed in the left side and the games are placed in the top side. Whenever a topic was represented in the game placed in the top side, a mark is placed representing the topic and the game it is covered in. In this table the topics that appear in the ACM Computer science curricula 2013 that have less representation in the games have been removed.

4.3.3 Google Play store games
From the information acquired after the analysis of the games that come from the Google Play store, the following topics have been highlighted as those who contain more representation inside the games. This information is mainly acquired by the programming concepts represented in the game section made in each of the analysis of the games available in appendix C. In table 3, the topics, that had more representation in the games, are placed in the left side and the games are placed in the top side. Whenever a topic was represented in the game placed in the top side, a mark is placed representing the topic and the game it is covered in. In this table the topics that appear in the ACM Computer science curricula 2013 that have less representation in the games have been removed.
**Table 2** Most covered topics by Steam store games

<table>
<thead>
<tr>
<th></th>
<th>7 Billion humans</th>
<th>Else Heart Break</th>
<th>Human resource machine</th>
<th>Shenzhen I/O</th>
<th>While True: learn()</th>
<th>Marvellous Inc.</th>
<th>Exapunks</th>
<th>Robots: create AI</th>
<th>Spacechem</th>
<th>TIS – 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expressions and assignments</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple I/O including file I/O</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Conditional and iterative control structures</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Functions and parameter passing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program comprehension</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program correctness</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Simple refactoring</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debugging strategies</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentaion and program style</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Table 3: Most covered topics by Google Play store games

<table>
<thead>
<tr>
<th></th>
<th>Lightbot</th>
<th>Gladiabots</th>
<th>Coddy</th>
<th>Algorithm City</th>
<th>Coding planets 2</th>
<th>Spritebox</th>
<th>game for kids</th>
<th>Human resource machine</th>
<th>Programming for kids</th>
<th>Coding planets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional and iterative control structures</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Functions and parameter passing</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Recursion</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

4.4 List of game design patterns

In this section, the different game design patterns used for representing the different programming concepts in the games is presented. It also contains a brief explanation on how the game pattern is presented in the games and how they are representing the different commonly represented topics mentioned in the section 4.2.

There might be several game patterns that represent the same topics but in a different way. This is due to the way of representing a topic that has each game, and therefore there might occur that one game represents one topic using one pattern but other uses a different one.

These game design patterns have been developed and identified by the author of this dissertation and inspired by the patterns available in the webpage of Björk (2019), where several game design patterns coming from several games have been identified for their later use.

4.4.1 Name: Block-based programming

One of the game design patterns commonly used in games for programming teaching is block-based programming. This pattern consists in making different blocks that represent certain actions inside the games. The purpose of this is to simplify programming and therefore, avoiding the different syntax mistakes that the players could make while at the same time they provide a system that uses programming for solving problems. These blocks connect with each other to form a program or programs, but for doing so they must be placed inside a function. Normally the players have access to at least one, representing the main function, but sometimes they can get access to other functions that for running them they should be placed inside the main function.
This pattern represents the assignments of the variables as well as the different actions that can be done through the game. It also is used for making conditionals and loops as well as structuring the program and writing comments to make it easily readable. Therefore, the topics that this pattern covers are expressions and assignments, conditional and iterative control structures and program comprehension.

Examples of the use of this pattern 7 billion humans or Human resource machine. These two games use the different blocks to represent actions that make the characters in the game move, grab things or operate with variables. They are also used to create conditionals and loops so certain actions are repeated or played when the conditional is fulfilled. This game design pattern is also found in while True: learn (), Robots: Create AI, Spacechem, Lightbot, Coddy, Algorithm city, Coding planets 2, Spritebox, Coding game for kids, Programming for kids and Coding planets.

4.4.2 Name: Step by step program execution
This game design pattern is based on giving the players the option to run their solution step by step or, as it is called in various programming environments, line by line. What this means is that for running every line of code that appears in the program, the players will have to click the provided button once per line of code. This button is an alternative one to the normal play button that will let the players search for a specific error in the program or to see what each line of the program does.

This pattern is mainly used for debugging purposes, where in a normal programming environment the programmers will have the possibility to put different breakpoints in the code, where it will stop running and let them see the outcome of it until that point. These lets programmers search for errors or points where the program does not behave as expected. The button that is provided inside the game is like this one, where it simulates a similar behavior. Therefore, the topic or topics this game design pattern covers are program correctness and debugging strategies.

Examples of the use of this pattern are 7 billion humans and Human resource machine. These games add a separate button next to the play button that will run the main program step by step. Normally, this button serves the players to see what a specific step in the program makes, useful for searching for errors in the program or for tracking strange behaviors of it.

4.4.3 Name: Visual Input and output representation
This game design pattern consists on making the games more interactable or visually represented. For making the player know what the task in the game is, the inputs of the different objects or variables that will take part in the process and the output where the results need to be placed are represented as a visible graphic inside the game. Sometimes it could be a post place or a hole where the objects need to be placed.

With this pattern, the objective in the game is to represent the different inputs and outputs that a program has in their functions or in the main program, as visual objects inside the game. This also represents the process these objects or variables take inside the game, like the place they come from, and the place where the results must go. In a programming environment this could be an external source that inputs data to the program, where the data
is treated and then outputted back to the main source or to other sources with a return. Therefore, the topic that this game design pattern covers is Simple I/O including file I/O.

Examples of the use of this pattern are TIS-100 or Spacechem where the input and output zones are visible to the player right after they enter to the puzzle. In this case they are represented as an arrow with their name on it. In most of the levels, the players will receive numbers that the players will have to operate with and deliver them to the output.

4.4.4 Name: Instruction-based programming
This game design pattern consists in basing the parts that involve programming inside the game in a procedural based programming language that simulates an assembler with their instructional approach. This means that the whole programming part is based around some instructions that, by passing them parameters they will do certain things. As an example of an instruction, the instruction mov is commonly used while controlling processors for moving data from a register to another position. These instructions are placed in the main program, where each of them will occupy a line of code, that will run in the order they are placed.

With this pattern, as with the pattern of block-based programming, the objective is to set an environment where the players can structure their programs. With this the players can simulate loops and conditionals, as well as assigning values to the registers available to them or doing operations with the different values. Therefore, the topics that this game design pattern covers are the expressions and assignments and conditional and iterative control structures.

As examples of the games that use this game pattern, Shenzhen I/O and Exapunks are based in an instruction-based programing approach. They use similar kind of instructions for making the programs of the circuits in Shenzhen I/O and the robots in Exapunks. Both games contain a manual explaining the use of the different instructions as well as examples on how to use them. These instructions will be placed in the program that the circuits or the robots will play for them to do the needed actions in the order specified. These are the tools given to the players to solve the different tasks they are given.

4.4.5 Name: Registers’ representation
This game design pattern consists in giving the players a way for storing data inside the games. This data storage simulates the registers a processor has for managing the different data used by the instructions. Normally these registers can store a singular piece of data in them at a time, something that is also represented in the games. These is done by allowing the players to store only one piece of data at a time, getting replaced every time they try to write a value in a used one.

With this pattern, the objective is to make a place where the players can store the data as if they were variables in the game. It is common that in this kind of games, the players will be required to do different mathematical operations, so the main variable type they will have access to is only one: numbers. Therefore, the topic this game design pattern covers is expressions and assignments.

An example of the use of this game pattern is seen in games like Shenzhen I/O, 7 billion humans, Marvellous Inc. and others where the players are provided with registers, whether it is in the components of the circuits like in Shenzhen I/O, in the workers themselves like in
7 billion humans or in consoles and the main character like in Marvellous Inc. These registers are used in the games for storing numbers that might be of interest for doing some mathematical operations. In some of the puzzles they tell the player to subtract the first number they will get to the third one they will be getting, so for doing so they need to store the first number for making the mathematical operation later when they get the number they need.

4.4.6 Name: Commentary writing

This game design pattern provides the players with a way of documenting their programs by writing different commentaries in the code. These commentaries will let them write explanations of the different parts of the code making them easy to understand. In games, the players are provided with different ways of writing these commentaries so they can place them in key parts of the code in order to explain what is happening in them.

With this pattern, the objective is to make the different programs easily understandable. For doing so there are many ways to present players with mechanics that let them write commentaries. This commentary writing can be presented by leaving blocks, that will let the players write things or make drawings that will not influence in the code, or by specifying a character that will represent the start of a commentary line inside the code. Therefore, the topic this game design pattern covers is documentation and program style.

As an example of the games that use this pattern, we have Else Herart.Break implements the option of writing comments in the programs of the different computers they can hack. In some of them they will already be able to see some comments made in the program they obtain explaining what some lines of code do or what the program does. In this game, this game design pattern is presented by letting the players write the character “#” at the start of the line where the players want to write the commentary. This pattern is also present in games like Marvellous Inc., Shenzhen I/O, Exapunks or TIS-100

4.4.7 Name: Hidden game programming challenges

This game design pattern is based in providing the players different challenges that are hidden to them until they finish the level, or something happens in the game. Normally these challenges are presented with certain extra objectives that will provide the players with rewards or with a mark in the level that shows that they completed all the challenges. This kind of things are not usually completed by everyone as they usually do not give anything else but self-satisfaction.

With this pattern, the objective is to provide a certain extra difficulty to the game that is not acquired directly in the levels. When we talk specifically for programming learning games, these challenges can be for optimizing the program solution they have made in various ways. Sometimes, the game will challenge the player to reduce the amount of code lines they needed for making their solution or making their program to run faster. Therefore, the topic covered with this game pattern is simple refactoring.

As an example of the use of this pattern, the games 7 billion humans and Shenzhen I/O use these hidden challenges to make a competition between the players. These challenges are shown at the end of each level where the statistics of how efficiently other players have solved the same level appears. In 7 billion humans, when completing these extra challenges,
a light in the elevator will be lighted but in Shenzhen I/O nothing will change, and no reward will be given.

4.4.8 Name: Program examples levels
This game design pattern consists in starting certain levels with code written on it. The code this level contains will sometimes explain how a new characteristic that is introduced into the game works or make a first introduction on how to proceed with the solving of the levels from that point.

With this pattern, the goal is to give an already made program to understand how the things work and, in some cases, to modify them in order to elaborate the working solution that will solve that level. Therefore, the topics that are covered with this game design pattern are program comprehension and documentation and program style.

As an example, this game design pattern is used in games like TIS-100, 7 billion humans and Human resource machine, where in some of the levels, normally the ones that present new mechanics, code will be already written inside them. This code will be there for the players to modify as they want or sometimes it will work by itself solving the puzzle but giving information to the player on how something works.

4.4.9 Name: Device hacking
This game design pattern consists in hacking devices that are available in the game. When hacking these devices what the players will be given is the actual program that was running inside that device. Sometimes, in the code of these programs the players are provided with comments on what does what and what is the real purpose of the program itself.

Within this pattern, the players are given programs to understand, with some examples explaining the things they can do in the game. Players can modify them if they want the device to do something else. Therefore, the topics covered with this game design pattern are program comprehension and simple refactoring.

As an example, this game design pattern is used in Else Heart.Break where the puzzles of programming appear when hacking the different devices available inside the game. These devices will sometimes have a complex program written that the players will have to understand in order to modify it. This game design pattern is also present in TIS-100 and Exapunks. This game design pattern belongs to the mentioned dark design patterns, as hacking is considered an illegal activity and therefore using it as a mechanic in the game could be considered something harmful for the players morality.

4.4.10 Name: Program output log
This game design pattern is based on presenting the players with the outcome of the program in form of text or other visual representation of it. This provides the players with feedback they can look to if they have errors in the program and do not know where the problem is.

With this pattern, the objective is to provide the players with feedback while running the program. With this information they can see the different actions that have been executed and where is the mistake that made them fail the level. Therefore, the topic covered with this game design pattern is debugging strategies.
As an example, this game design pattern is used in *Robots: create AI*, where there exists a button available to the players that will open the code log tab. This will show up the movements and the decision the robot has made during the running of the program the specified number of times. This game design pattern is also present in games like *7 billion humans*, *Human resource machine* and *Marvellous Inc.* where in some cases the players will be given certain feedback of mistakes that made them fail the level.

**4.4.11 Name: Resource limitation**

This game design pattern is based on limiting the resources available to the players. This will make the players need to use the limited resources they have access to by reusing parts of the program with tools like functions. The functions available in the games allow players to repeat several actions that are need inside the levels more than one time. For saving space in the main function, that might have space limits, the actions can be placed inside a function and this function can be called inside the main one as many times as needed.

With this pattern, the objective is to optimize the different programs by making the players use all the tools available to them. If by making a function that will make the actions that need to be repeated through the program will optimize the code, the players will be forced to do that to end up with a viable result. Therefore, the topics covered with this game design pattern are functions and parameter passing and recursion.

As an example, this game design pattern is used in games like *Lightbot* where if the players do not use the different functions available for making loops, certain levels will not have enough spaces and therefore, they will be impossible to solve. This game design pattern is also available in *Programming for kids – Learn coding*, *Lightbot: Code Hour*, *Coddy: World on algorithm Free*, *Algorithm City: Coding game for kids*, *Coding planets 2*, *Spritebox: code Hour and Coding game for kids – Learn to Code with Play*

**4.5 General outcomes**

Game availability during the development of this dissertation has been a problem. In Steam, the games that the community tagged to have programming concepts in them, were in some cases based in hacking devices where no programming concepts were represented inside them. Therefore, the game availability was lower than what it was thought to be at the start of this study. Another problem that appeared in the process of this study, is that by looking at the stats of the games that appear in the page “playtracker.net/insight”, the popularity indicator of games that had programming concepts inside them was really low, and the play rate and the completion percentages were both low as well.

However, the existing problem with Google Play store games for programming learning was that by using the keywords mentioned in section 3.1.1, the applications that show in the search were mostly apps for learning programming that had nothing to do with games, as well as games that focused in teaching kids how to program.

In the graph of figure 2 the classification of the different topics by the number of games that represent them is shown. This shows that only the first 5 topics that appear in it, conditional and iterative control structures, functions and parameter passing, debugging strategies, program correctness and simple I/O and file I/O, have representation in half of the 20 games or more. This shows the lack of representation various topics have inside these kinds
of games, what makes them, in a way, not viable for their use in educational environments without complementing them with explanations outside of the game.

However, there are some games that innovate by introducing ways of solving puzzles and explanations, like in Where True.learn(), in a more immersive way as well as providing links to sources containing information of the corresponding concept in real programming environments.

Along with this, several game design patterns were identified in this study. With this game design patterns, several similarities in the way that the games represent the same concepts are identified. Even if the topics that a game represents variates from one to other, those games who focused in the same ones used a similar game design pattern to represent that in their game. Therefore, the ones identified in this study serve as tools for future game development.

From all the games that have been analyzed, the most common genre between them was the puzzle-based game. This genre consists in basing the games around different levels or mechanics that make the players to solve different puzzles with certain mechanics. As these games were based around programming, the most common thing between them is that the puzzles consist in making programs that when executed should complete certain tasks.

From the dark patterns explained in the section 3.2, no game design patterns that cause negative experiences in the players have been found inside the games used for the analysis. However, from the design patterns that belong to the “Shades of gray”, some of the games encourage the hacking behaviour and the stealing of information from others in order to obtain money or other benefits.
5 Conclusions

In this chapter, the different conclusions that derived from the study made in this paper are explained. First, a summary of the different results and conclusions acquired from the study, and after a discussion about other relevant work within the field ending with a possible future works section. Within this section, the answer to the main research question “which of the fundamental programming concepts are represented in commercial games for programming learning and what game design patterns are used for representing them?” is explained.

5.1 Summary

Serious games for programming learning have evolved through the years and an immense of research has been done in that sense. But still, the availability of the games is still very low and most of them belong to different research projects where the games are hard to find. But regarding to the most popular stores, the games that are available are still not popular within the gaming community, where even if they have thousands of owners, the play rate is very low, and the percentage of game completion of each user is also very low.

Despite that, these games have still a long way to go until they become good solutions for learning programming. From the topics that the ACM Computer Science Curricula 2013 states as fundamental for a course of computer science, these games do not cover near half of them by themselves. This shows the lack from which these games suffer nowadays, what makes them hard to be introduced in education environments.

From these games, it can also be deduced that there is some relation between the topics the cover inside the games and the way they show them with their game design patterns. As an example, those games, who focused on a more complex way of teaching programming, were more used to represent the concepts with a more complex UI, making the players need to code the different programs with a pseudocode, specifically made for the game, or simulating an existing one. However, games with a simpler approach, are more used to make a block-based programming system where the syntax is not a problem for the players, simplifying in some way the mechanics. Like these game patterns, other similarities were found inside the games for representing the same topics, which are listed in previous sections for possible future use for whoever interested to develop programming learning serious games.

Nevertheless, there was a huge similarity in all the games which was that was that mostly all of them were based on puzzle-solving solutions with levels inside the game containing the mentioned puzzles. It is true that there are exceptions like Else Heart.Break() which is based on an history driven game with point and click mechanics where the main objective is to hang around the city. But even in that game that was not mainly based around solving puzzles, the programming part was, in some kind, a puzzle solving mechanic where you had to write the program of different devices to make them do something you need for that situation, or to unblock a door you need to get through.

The only difference found between the games searched in Steam and in Google Play is that the Google Play games tended to be more simplistic and childish and generally were advertised as programming games for kids. This results in a similar representation of the topics as the ones used in the Steam store but in a more simplified way. Where in 7 billion
humans the block-based programming had words inside them that described the action, in Lightbot the blocks had symbols like arrows for representing the action and the direction at the same time. This simplifies the understanding of the mechanic of the game, and its directly associated with a certain action the character will do with that.

As a final result, in this paper has been proven that there exist topics for which a lot of commercial games exist, like the conditional and iterative control structures topic that is covered in almost every game, but there also exist topics that are not even represented in any game like abstract data types or several other data related topics that have no games representing them in any way.

5.2 Discussion

In the same way that has been made with this dissertation, there are two other research papers documenting different games for programming learning. These papers have been mentioned throughout the paper, the first one is “A review of games designed to improve introductory computer programming competencies” by Vahldick et al. (2014) and the second one is “A review of serious games for programming” by Miljanovic and Bradbury (2018). These two papers are mainly based in researching and analyzing games that appeared in research papers as solutions for programming learning, but they also covered some commercial games they considered important for their research.

The results shown in these two papers differ a lot from the ones that appear in this one, showing a high number of games representing themes that commercial games have not covered at all or are very briefly introduced or mentioned. One of the huge important differences that can be seen is the lack of problem-solving strategies explained inside commercial games, as giving explanations with text inside the games might be tedious and time consuming for the players, where in the research paper games it was a primary focus in a lot of them, as they focused more in the teaching aspect. With this information we can deduce that even if in researches certain topics have been chosen as interesting ones, it has not been the same in the commercial market.

As a side note, in research paper games there exist several game design patterns that have been used like the concreteness fading used in the game Reduct (Arawjo, Wang, Myers, Andersen and Guimbretière, 2017), researched and explained in their paper but that have not been used in any of the commercial games reviewed in this paper. Maybe, the problem is that, even if research in the field is being done, the different game design approaches that are taken in the different solutions for programming learning are not expanded as working solutions for others to use. Therefore, as the developers do not know that some game design patterns, that have been proven to work, exist, they will not use them, making the research practically useless.

There is also a lack of a planification phase of programming involved in the analyzed games. Robins, et al. (2003) wrote in their study that one of the main differences between experts and novices is that experts normally tend to spend more time in the planification phase, making various schemas and design strategies before starting to write the actual program to organize how the result should be. This is an important task in the programming learning process as it these phases provide a comprehension of how a program works inside a computer that makes the novices feel more confident while programming and therefore
making less mistakes. It also provides with a more organized and well written code, that makes it easier to understand.

5.3 Future Work

There is a lot of research still missing in the field of serious games for programming learning. The solutions that are available nowadays, unfortunately not many, focus in the representation of the same concepts through them, normally simplifying them. Even though, this seems like a downside, simplifying programming concepts could help in solving the current problem of the high entry barriers in computer science courses.

Nevertheless, some research would be desirable to be made talking about the different advantages and disadvantages of making simplified programming environments for programming learning versus more modern programming environments like Visual Studio, a programming environment made by Microsoft. This is one of the topics that was not covered by any of the games that appeared in the analysis, modern programming environments. This is a relevant topic within the programming learning field, as nowadays there exist several environment solutions, normally free to use, that provide programmers with tools to help them while programming. These tools range from debuggers, to autocompletion of lines of code by predictive writing. These environment solutions should be presented or explained in some way to the players that want to continue with their learning process after playing the game.

With the topic explained above, several others were highlighted as those who had little or no representation inside the games. From the main theme of fundamental data structures, none of the topics was chosen to be represented in the games other than the arrays, and only in some of them. Some of the games had representation of registers where the players were able to store values for their later use, but despite that, the concept of an array was not explained in the majority of the games. These data structures and the different strategies for choosing the appropriate one is important while programming, as not choosing the appropriate one might result in optimization problems or other mistakes that usually novices make while programming.

Along with this main theme, the “algorithms and design” theme was also the other one that had none or very little representation inside the games. This theme is based in on topics that relate concepts presented to computer science students before they handle writing programs. They present different explanations of what an algorithm is and how is used, as well as problem solving strategies useful for analyzing a problem before writing the program that will solve it. The final topic is about design concepts and principles where students are taught a way to organize their code while programming, and how to achieve knowledge by using other programs code as examples.

These two main themes that were not represented inside the games should be the focus of the next researches for addressing if it is possible to fully implement the topics describe in the ACM Computer Science Curricula 2013 as fundamental for a computer science course. However, not all of the serious game solutions need to cover all the topics mentioned in this paper, but it could be that several solutions cover different topics using a different approach in every one of them.
Also, the game genre that was more commonly used in the analyzed games was the puzzle-based games. Research in possibilities of making games that belong to other game genres should be made to see the capabilities programming based games have and what problems does it present while making this kind of games.
References


Marijan, 2019, playtracker insight, <https://playtracker.net/insight/>


Appendix A - Game list

Steam

1. Baba is you
2. TIS-100
3. Opus magnum
4. Spacechem
5. Shenzhen I/O
6. Exapunks
7. Silicon zeroes
8. Human resource machine
9. Infinifactory
10. Gladiabots
11. 7 billion humans
12. Prelogate
13. Hacknet
14. [The Sequence]
15. Quadrilateral Cowboys
16. MHRD
17. Prime Mover
18. Great Permutator
19. While True: learn ()
20. Duskers
21. Screeps
22. Super Markup Man
23. Drunken Wrestlers 2
24. Else Heart.Break()
25. Mainlining
26. OCTOPTICOM
27. MuX
28. Senalux
29. Cool Headed
30. CompliKATed
31. MarionetteAI
32. Rogue Bit
33. Algo Bot
34. Cyber Sentinel
35. Jump, Step, Step
36. Adventure Land – The code MMORPG
37. RoboSnakes: Core Wars Legacy
38. LogicBots
39. Hackmud
40. Debugger 3.16: Hack’n’Run
41. Glitchspace
42. Marvellous Inc.
43. Planetoid Pioneers
44. Glyphs Apprentice
45. God is a Cube: Programming Robot Cubes
46. Hack 'n' Slash
47. Code World
48. Hardware Engineering
49. Algotica Iterations
50. CodeSpells

Steamspy.com

1. Hacknet -> 2.000.000 – 5.000.000
2. SpaceChem -> 500.000 – 1.000.000
3. Else Heart.Break() -> 200.000 – 500.000
4. Human Resource Machine -> 200.000 – 500.000
5. Hack ‘n’ Slash -> 200.000 – 500.000
6. TIS-100 -> 200.000 – 500.000
7. Infinifactory -> 200.000 – 500.000
8. Opus Magnum -> 100.000 – 200.000
9. While True: learn () -> 100.000 – 200.000
10. Drunken wrestlers 2-> 100.000 – 200.000
11. Shenzhen I/O -> 100.000 – 200.000
12. Baba is You -> 50.000 – 100.000
13. Duskers -> 50.000 – 100.000
14. Quadrilateral Cowboy -> 50.000 – 100.000
15. Glitchspace -> 20.000 – 50.000
16. Screeps -> 20.000 – 50.000
17. Great Permutator -> 20.000 – 50.000
18. Algotica Iterations -> 20.000 – 50.000
19. Exapunks -> 20.000 – 50.000
20. Hackmud -> 20.000 – 50.000
21. I hate this game -> 20.000 – 50.000
22. 7 Billion Humans -> 20.000 – 50.000
23. Mainlining -> 0 – 20.000
24. Silicon Zeroes -> 0 – 20.000
25. Prelogate -> 0 – 20.000
26. Algo Bot -> 0 – 20.000
27. [The sequence] -> 0 – 20.000
28. Robots: create AI -> 0 – 20.000
29. Marvellous Inc. -> 0 – 20.000
30. Jump, Step, Step -> 0 – 20.000
31. Cyber sentinel -> 0 – 20.000
32. Super Markup Man -> 0 – 20.000
33. Senalux -> 0 – 20.000
34. Prime Mover -> 0 – 20.000
35. Planetoid Pioneers -> 0 – 20.000

Playtracker.net/insight

1. Hacknet -> 2.8 M -> 34 popularity
2. SpaceChem -> 1.6 M -> 21 popularity
3. Else Heart.Break() -> 921 K -> 12 popularity
4. TIS-100 -> 859 K -> 11 popularity
6. Infinifactory -> 689 K -> 9 popularity
7. Hack ‘n’ Slash -> 568 K -> 7 popularity
8. Great Permutator -> 358 K -> 5 popularity
9. Mainlining -> 324 K -> 5 popularity
10. Shenzhen I/O -> 219 K -> 3 popularity
11. [The sequence] -> 217 K -> 3 popularity
12. Algotica Iterations -> 189 K -> 3 popularity
13. Opus Magnum -> 183 K -> 2 popularity
14. Prelogate -> 156 K -> 2 popularity
15. Glitchspace -> 143 K -> 2 popularity
16. Baba is You -> 108 K -> 3 popularity
17. While True: learn () -> 96 K -> 1 popularity
18. Cyber sentinel -> 93 K -> 1 popularity
19. Duskers -> 82 K -> 1 popularity
20. Marvellous Inc. -> 80 K -> 1 popularity
21. Screeps -> 58 K -> 1 popularity
22. Quadrilateral Cowboy -> 56 K -> 1 popularity
23. Exapunks -> 54 K -> 1 popularity
24. Hackmud -> 45 K -> 0 popularity
25. 7 Billion Humans -> 45 K -> 1 popularity
26. Prime Mover -> 14 K -> 0 popularity
27. Silicon Zeroes -> 13 K -> 0 popularity
28. Super Markup Man -> 12 K -> 0 popularity
29. Algotica Iterations -> 10 K -> 0 popularity
30. Robots: Create AI -> 6 K -> 0 popularity

Play store

1. Lightbot: Code Hour -> 1.000.000+
2. Circuit Scrambler -> 1.000.000+
3. Gladiabots -> 500.000+
4. Coding planets -> 100.000+
5. Tynker – Learn to code -> 100.000+ (Discarded due to not being available to
download and no good information source was found)
6. Coddy: World on algorithm Free -> 100.000+
7. Algorithm City: Coding game for kids -> 50.000+
8. Bit by Bit – Programming Game -> 50.000+
9. Coding planets 2 -> 50.000+
10. SpriteBox: Code Hour -> 50.000+
11. Coding game for kids – Learn to Code with Play – 50.000+
12. Human resource machine -> 10.000+
13. Robotizen: Kid learn Coding Robot -> 10.000+
14. Lobot - Robot Programming -> 5.000+
15. Programming for Kids – Learn Coding -> 5.000+
Appendix B - Initial scanning phase

This part of the game analysis will contain the description of the games as well as an explanation of the main objective the player has inside them. The description will be extracted directly from the store and the explanation of the main objective will be made by looking into the videos and photos they provide as well as by looking into YouTube videos of the games.

Steam games

Hacknet

https://store.steampowered.com/app/365450/Hacknet/

Description

Hacknet is an immersive, terminal-based hacking simulator for PC. Dive down a rabbit hole as you follow the instructions of a recently deceased hacker, whose death may not have been the accident the media reports.

Objective

Realize different tasks that you will receive via the email service implemented inside the game. These tasks will consist in making you write certain commands in the command prompt that appears in the right part of the screen and connect to different nodes in the same network. Sometimes you will also have to execute scripts to make things by just writing the name of the script in the console (no programming required).

Overall programming level

Very low, only commands for the console.

Spacechem

https://store.steampowered.com/app/92800/SpaceChem/

Description

Zachtronics is back with an ambitious new design-based puzzle game. Take on the role of a Reactor Engineer working for SpaceChem, the leading chemical synthesizer for frontier colonies. Construct elaborate factories to transform raw materials into valuable chemical products!

Objective

Generate circuits in order to transform raw materials into the chemical product required in the level. You have access to different tools to create this circuits in the most efficient way possible. You have access to loops, conditionals, input and output controls and something like functions or threads.

Overall programming level
Medium. Based on programming logic to generate loops that will create chemicals, concurrency involved inside the game for the timing of the factories. No code writing involved.

**Else Heart.Break()**

https://store.steampowered.com/app/400110/Else_HeartBreak/

**Description**

A game about friendship, love and technology in a place where bits have replaced atoms.

**Objective**

A story driven game where the main objective of the player is to solve the different puzzles of the city to continue with the story the game is trying to tell. Inside the game you have access to different puzzles containing programable objects or code parts you can have access to and modify to fix or make things.

**Overall programming level**

Medium-High. You can modify the code of programs that will make different machines modify their behaviour. You basically hack machines.

**TIS-100**

https://store.steampowered.com/app/370360/TIS100/

**Description**

TIS-100 is an open-ended programming game by Zachtronics, the creators of SpaceChem and Infinifactory, in which you rewrite corrupted code segments to repair the TIS-100 and unlock its secrets. It’s the assembly language programming game you never asked for!

**Objective**

The players must solve certain puzzles in order to repair the TIS-100 by rewriting the corrupted code segments. To do this you have to write little programs inside blocks with a pseudo programming language. You are given a user manual with all the details and the commands you can use to write these little programs. The objective of these little programs is to work with each other to fix this corrupted code segments and solve the puzzle.

**Overall programming level**

Medium. The pseudo programming language used inside the game is like an assembler, procedural language based on instructions that get executed line by line controlling the registers of the processor of a computer.

**Human resource machine**

https://store.steampowered.com/app/375820/Human_Resource_Machine/

**Description**
Program little office workers to solve puzzles. Be a good employee! The machines are coming... for your job. From the creators of World of Goo and Little Inferno.

**Objective**

You must solve different puzzles by programming the behaviour of the different workers you have access to. For programming the workers behaviour, they provide you with different blocks of code that will interact with each other. Therefore, you have to organize these blocks in the correct order to make them do what you want and solve the various puzzles.

**Overall programming level**

Medium-High. The connection of the blocks is based in a procedural program language. You have to basically write procedural programs but using blocks instead of code.

**Infinifactory**

[https://store.steampowered.com/app/300570/Infinifactory/](https://store.steampowered.com/app/300570/Infinifactory/)

**Description**

Infinifactory is a sandbox puzzle game by Zachtronics, the creators of SpaceChem and Infiniminer. Build factories that assemble products for your alien overlords and try not to die in the process.

**Objective**

The player has to solve different puzzles with the given tools. Each puzzle will require the players to create a small factory with the blocks they have access to. This factory should produce a concrete amount of the product they want without the factory failing.

**Overall programming level**

Low-Medium. Making the factory requires the player to think of the actions the materials will have to follow to create the different materials. You have access to items that simulate conditionals, loops and other programming like concepts.

**Hack ‘n’ Slash**

[https://store.steampowered.com/app/246070/Hack_n_Slash/](https://store.steampowered.com/app/246070/Hack_n_Slash/)

**Description**

Hack ‘n’ Slash is a puzzle action game about hacking -- reprogram object properties, hijack global variables, hack creature behavior, and even rewrite the game’s code. The only way to win is not to play...by the rules!

**Objective**

It is an action like game where you have to fight through different mobs and you have the ability to change the different variables of the game, like the attack, the life, the behaviour of
the creatures etc.... You will also have to solve puzzles with this same mechanic of “hacking” the game through modifying the different block behaviour or the game itself.

**Overall programming level**

Medium. You have access to variable management and sometimes to the representation of the code of certain blocks inside the game by graphical objects. Sometimes you can modify these things to make them do what you want.

**Great Permutator**

[https://store.steampowered.com/app/319270/Great_Permutator/](https://store.steampowered.com/app/319270/Great_Permutator/)

**Description**

Great Permutator is a puzzle. Hard and tricky. One in which you will have to build a complex system of conveyors and various devices so that the colored boxes can travel across the factory and arrive at right places and in right order. If you are into clever engineering puzzles, this game is for you.

**Objective**

The players have to solve different puzzles where the objective is to guide the different blocks of colour to their final destination. You have access to different action blocks that will classify the blocks and guide them.

**Overall programming level**

Low-Medium. You have access to blocks that will act as conditionals, but no other programming concepts represented inside the game.

**Mainlining**

[https://store.steampowered.com/app/454950/Mainlining/](https://store.steampowered.com/app/454950/Mainlining/)

**Description**

After MI7 has been hacked, your primary goal is to bring a cyber-criminal network to justice. Played entirely from within the environment of your computer's desktop, Mainlining is a thrilling point and click hacking adventure that features a gripping narrative which will have you hooked from the very start.

**Objective**

The players have to fulfil certain tasks by using their computer as the only tool. These tasks will be about investigation and finding certain useful information for continuing with the game. Some of them will be puzzles they will have to solve where some will be about using the command console and hack things with it.

**Overall programming level**
Low. In some of the puzzles you will have access to a command console to hack some things with the use of some simple commands.

**Shenzhen I/O**

[https://store.steampowered.com/app/504210/SHENZHEN_IO/](https://store.steampowered.com/app/504210/SHENZHEN_IO/)

**Description**

BUILD CIRCUITS. WRITE CODE. RTFM.

**Objective**

Solve different puzzles by building different circuits and programming them to do certain things. You have limited space for placing the components, therefore limited resources to solve each of the puzzles.

**Overall programming level**

Medium – High. This game is based on a pseudo code like an assembler or circuit instructions. You manage the values with this instruction-based language to build circuits that will solve the various puzzles.

**[The sequence]**

[https://store.steampowered.com/app/454320/the_Sequence/](https://store.steampowered.com/app/454320/the_Sequence/)

**Description**

[the Sequence] is a logic construction puzzler. Let your mind be your guide: Lay down seven different kinds of modules as your roadblocks to get you (portrayed as a binary data point) to a sweet destination. Apply an out-of-the-box thinking to get you to the next level!

**Objective**

The game provides a series of levels that the players have to complete. In the levels they have access to certain blocks that will let them guide the data that represents yourself to your destination.

**Overall programming level**

Low. For solving the puzzles, you have different blocks that by placing them in the board will guide the data in some way.

**Algotica Iterations**

[https://store.steampowered.com/app/593330/Algotica_Iterations/](https://store.steampowered.com/app/593330/Algotica_Iterations/)

**Description**

Algotica is an educational adventure-puzzle game about programming, along with a story that breaks the fourth wall. If you wanted to start learning programming but it seems too
difficult to understand, you don’t have enough time, or you’re just not sure that you need it - this game is a great place to start.

**Objective**

The players must write certain commands available in a certain order to solve the different puzzles that appear in the game. These commands can be placed as many times as required but the program, they write with them will be played in a loop when stepping in a certain type of blocks. For solving these puzzles there exist certain mechanics the players have to interact with.

**Overall programming level**

Low – Medium. The players have access to a main function like panel that will play in a loop. In this main function they have to use the commands available in a certain order to complete the puzzles. They don’t have access to conditionals directly, but the game makes use of them by not letting the players succeed if they don't push a certain button or collect certain objects.

**Opus Magnum**

[https://store.steampowered.com/app/558990/Opus_Magnum/](https://store.steampowered.com/app/558990/Opus_Magnum/)

**Description**

Opus Magnum is the latest open-ended puzzle game from Zachtronics, the creators of SpaceChem, Infinifactory, and SHENZHEN I/O. Design and build machines that assemble potions, poisons, and more using the alchemical engineer’s most advanced tool: the transmutation engine!

**Objective**

The players are required to create a system with robot arms that will create the chemical component they are required in each level. You have access to certain blocks that will let you program the behaviour of the different arms, controlling the time of each action and ending with a final system that will produce the amount of the chemical needed without stopping.

**Overall programming level**

Medium. You have access to the programming of the different robot arms by using certain blocks that let you program conditionals and other actions. You also have control on the timing of the actions.

**Prelogate**

[https://store.steampowered.com/app/332830/Prelogate/](https://store.steampowered.com/app/332830/Prelogate/)

**Description**

Prelogate is a logic puzzle game. Solve puzzles by redirecting laser beams, mixing them and controlling them with logic gates.
Objective
The game provides different levels to be solved. In each level the players have to guide the lasers to their destination by placing different blocks that make different things.

Overall programming level
Low – Medium. Certain blocks available inside the game will represent logic gates to simulate conditionals and redirect the lasers.

Glitchspace
https://store.steampowered.com/app/290060/Glitchspace/

Description
Glitchspace is about reprogramming the game to solve puzzles. Trapped in an abstract cyberspace world, your way forward uses visual programming to edit environmental geometry, whilst unlocking key programming concepts along the way.

Objective
The players will have to solve different puzzles by modifying the program of certain objects in the game and act as needed in that situation. You will have access to modify certain variables or behaviours of the items to suit your needs.

Overall programming level
Medium. You can modify the different variables of the objects in the game as well as their behaviour with the player. You can also modify their state and have access to conditionals and loops to modify their actions.

Baba is You
https://store.steampowered.com/app/736260/Baba_Is_You/

Description
Baba Is You is a puzzle game where the rules you have to follow are present as blocks you can interact with. By manipulating them, you can change how the game works, repurpose things you find in the levels and cause surprising interactions!

Objective
The players are required to solve different puzzles, normally by getting a flag that appears in all the levels. For doing so they have to move certain blocks to change the state of objects that interact with the player. Sometimes it could be changing the state of the wall to be able to walk through it or setting the flag to give the state of winning the level.

Overall programming level
Low – Medium. It has different strategies for setting variables through the game and change the state of objects inside the game, but no use of conditionals or loops or other programming concepts.

**While true: learn()**

[https://store.steampowered.com/app/619150/while_True_learn/](https://store.steampowered.com/app/619150/while_True_learn/)

**Description**

You're a machine learning specialist who makes neural networks, but your cat seems to be better at it. Now you must solve puzzles to build a cat-to-human translation system (who knows what else this cat is capable of!). Earn a fortune, buy kickass cat outfits and learn how machine learning really works!

**Objective**

The players must develop a system that translates cats’ language into human language. With this main objective you will have to solve different puzzles to make money and advance in the development of this system.

**Overall programming level**

Medium – High. This game is based in machine learning where you start collecting data and processing it in different ways. You will write your programs by dragging and dropping blocks into the main part and connecting them to work with each other. You have access to conditionals, loops and several other actions. It also mentions the term algorithm through the game, as well as other programming concepts like optimization.

**Cyber sentinel**

[https://store.steampowered.com/app/485970/Cyber_Sentinel/](https://store.steampowered.com/app/485970/Cyber_Sentinel/)

**Description**

Experience the hacker's side of cyberpunk in Cyber Sentinel! This design-based puzzle game features a complete visual programming kit for designing your own viruses.

**Objective**

The game provides the players with different levels to complete. The players have to program the behaviours of the blocks that appear in each level to guide them to their destination. To write the programs they have access to different actions that will get executed when running the program. Each block has 4 different colours, meaning that if the block touches a field with one of that colours it will execute the action that matches that colour.

**Overall programming level**

Medium. You can write programs with the block provided inside the game. At the same time all these blocks acts as conditionals as the following actions will match the colour of the block that activates them. You can also simulate loops.
**Duskers**

[https://store.steampowered.com/app/254320/Duskers/](https://store.steampowered.com/app/254320/Duskers/)

**Description**

Pilot drones into derelict spaceships to find the means to survive and piece together how the universe became a giant graveyard.

**Objective**

Control the different drones with the console available to the player and survive in the space.

**Overall programming level**

Low. You have access to the commands used to control the different drones through the command console.

**Marvellous Inc.**

[https://store.steampowered.com/app/827940/Marvellous_Inc/](https://store.steampowered.com/app/827940/Marvellous_Inc/)

**Description**

Marvellous Inc. is a programming puzzle game, where you program robots to solve tasks.

**Objective**

The players must program robots to solve certain tasks. These tasks will be provided in every level where in some of them they will be required to manage some variables or make some mathematical operations within the program.

**Overall programming level**

Medium – High. The players have to write the program of the robots to complete the tasks. They have access to a pseudo code similar to an assembler programming language. You have several instructions available that will make certain actions, like add, jmp, drop, etc.... It is like how processor manage the information between registers.

**Screeps**

[https://store.steampowered.com/app/464350/Screeps/](https://store.steampowered.com/app/464350/Screeps/)

**Description**

Screeps is an open source MMO RTS sandbox game for programming enthusiasts, wherein the core mechanic is programming your units AI. You control your colony by writing JavaScript which operates 24/7 in the single persistent world filled by other players on par with you.

**Objective**
The players must program the artificial intelligence of the different creeps that simulate their workers and characters in the game. This will set their behaviour through the game that can let them work, fight or do other sorts of things. The players must make them survive to the world by themselves with the AI they have programmed.

**Overall programming level**

Low. The players must program the artificial intelligence of the different creeps, that are the creatures that live in the game. This will set their behaviour in the different situations. To program the AI, they will have to use the program language known as JavaScript. The scripts are programmed outside the game so the game itself doesn’t contain programming in it.

**Quadrilateral Cowboy**

[https://store.steampowered.com/app/240440/Quadrilateral_Cowboy/](https://store.steampowered.com/app/240440/Quadrilateral_Cowboy/)

**Description**

Quadrilateral Cowboy is a single-player adventure in a cyberpunk world. Tread lightly through security systems with your hacking deck and grey-market equipment. With top-of-the-line hardware like this, it means just one thing: you answer only to the highest bidder.

**Objective**

The players will act as a system hacker that will try to steal a bunch of money or things from others by accessing to different locations hacking the doors and things that get in its way. They will use certain commands to hack the different doors and systems to achieve what they want.

**Overall programming level**

Low. You have access to different commands that you will have to execute in several occasions to hack the various systems. You don’t have access to any coding or anything neither to the use of conditionals or loops or other programming concepts.

**Exapunks**

[https://store.steampowered.com/app/716490/EXAPUNKS/](https://store.steampowered.com/app/716490/EXAPUNKS/)

**Description**

The year is 1997. You used to be a hacker, but now you have the phage. You made a deal: one hack, one dose. There’s nothing left to lose... except your life.

**Objective**

The players must code the program that different robots will execute in each level. To make these programs they have access to a pseudo language explained in manual inside the game. This pseudo code has similar instructions to those used in an assembler.

**Overall programming level**
Medium - High. The players have access to a pseudo code that uses assembler like instructions. They must write programs for each of the robots available inside every level. These programs will run at the same time, so the complexity exist in coordinating all the robots inside the puzzles.

**Hackmud**

https://store.steampowered.com/app/469920/hackmud/

**Description**

hackmud is a text-based multiplayer hacking simulator for Intel-compatible personal home computers. Crack protected systems and solve puzzles as you explore the abandoned internet of the future. Write scripts to protect your winnings and trick other players. Lose everything. Get it back.

**Objective**

The players are initiated with a single player tutorial where they must fulfil the different tasks they are given by a group of hackers. The game itself is a big command console where players will be required to use different functions and commands to complete the tasks. These functions will accept parameters depending the situation. After this single player tutorial is finished, they get into the multiplayer world where you can act as a hacker by stealing other players money or doing other jobs.

**Overall programming level**

Low - Medium. The players have access to different commands in the command console, as well as executing functions that with the correct parameters will do what they need. There is also the option in the multiplayer to create their own scripts by using JavaScript and importing them, but this could be taken as an external programming and not something you do inside the game.

**7 Billion Humans**

https://store.steampowered.com/app/792100/7_Billion_Humans/

**Description**

Automate swarms of office workers to solve puzzles inside your very own parallel computer made of people.... a thrilling follow-up to the award-winning Human Resource Machine. Now with more humans!

**Objective**

The players must write the behaviour of the workers by connecting blocks that represent certain actions. These blocks also can represent conditionals and loops.

**Overall programming level**
High. The players must write the programs of the workers by using blocks of actions. These blocks let you simulate loops and conditionals as well as variable management. This is made to fulfil certain tasks within the levels of the game.

**Prime Mover**

https://store.steampowered.com/app/693700/Prime_Mover/

**Description**

Retrace the path of the Prime Mover to uncover the story of the Byte of Burden! Prime Mover is an open-ended puzzle game about designing circuit boards.

**Objective**

The game provides the players with different puzzles representing circuits to be solved. For completing the circuits, they have to make different mathematic operations to guide the numbers asked by the level into the final destination desired.

**Overall programming level**

Medium. The players have access to circuit logic. This logic is like programming logic where they have access to conditionals for guiding different numbers to their destination. For doing so they have to make different mathematical operations with the blocks provided in the game.

**Silicon Zeroes**

https://store.steampowered.com/app/684270/Silicon_Zeroes/

**Description**

Build hardware. Solve puzzles. From the creator of Manufactoria, a new open-ended puzzle game set in Silicon Valley's first startup.

**Objective**

The players are required to perform certain mathematical operations to solve the different puzzles provided inside the game. These puzzles will be presented as tasks in the levels the game contains. For completing them they must guide the result to its destination after making the mathematical operations with the blocks provided in the game.

**Overall programming level**

Low – Medium. The players have access to blocks that represent certain mathematical operations. They must guide the numbers into several blocks to end with the result desired in its destination.

**Algo bot**

https://store.steampowered.com/app/286300/Algo_Bot/

**Description**
Algo Bot is puzzle game that takes place deep inside the Europa, a pan-galactic colonisation ship, where a recycling mission goes horribly wrong. Use your algorithm special skills to command Algo Bot, a service droid, and help PAL, a cantankerous line manager, to contain the crisis.

Objective

For advancing in the levels you must program the behaviour of a robot. This robot will have to reach the end of the level by interacting with several things in each level. For doing so, you have access to certain instructions that when placed in the main program will run action by action until it finishes.

Overall programming level

Medium. For making the program you have access to different instructions that will make the robot move or turn. It also gives the option for making functions and run them several times inside the main program.

Super Markup Man

https://store.steampowered.com/app/502210/Super_Markup_Man/

Description

Super Markup Man is the ultimate fun way to practice web development and learn the basics of HTML and CSS. No coding skills needed!

Objective

The players get to control different characters in the game. They will have to order the different blocks that appear in the levels to fix the HTML and CSS code they represent. When putting the things in the correct order, the representation of the web page that appears in the right side will change accordingly.

Overall programming level

Medium. Different representations of HTML and CSS code appear inside the game.

Robots: create AI

https://store.steampowered.com/app/676620/Robots_create_AI/#app_reviews_hash

Description

Robots: create AI is a puzzle programming game.

Objective

The game provides several levels for the players to solve demonstrating their programming skills. For completing the puzzles, they must write the program that will make the main character move and get to the final field tagged with a flag. For doing so, the players are
provided with blocks that will represent certain actions the robots will execute. All this block should be placed in a specific order to complete correctly each puzzle.

**Overall programming level**

Medium – High. The different blocks represented inside the game let the players program loops and conditionals inside the program in order to complete the different puzzles.

**Google Play store games**

**Lightbot: Code Hour**


**Description**

Lightbot - Code Hour is meant to introduce kids who have no experience whatsoever programming and is all-ages friendly! That means anyone anywhere from K to 12 can play, have fun and learn real programming logic!

**Objective**

The objective in this game is to control a robot by writing the program that will make it move. The robot needs to get to the blue square that appears in each level to complete it. For doing so the players have access to several blocks that correspond to certain actions and a main function that will execute the actions placed in the order desired.

**Overall programming level**

High. You have access to the main function that will execute the actions in the order the players place them. The main function has a limited amount of spaces to place blocks, so you must manage them in the correct way. It also gives you access to various functions through the levels.

**Circuit Scrambler**


**Description**

Circuit Scramble throws you in to the world of circuit-based logic that drives real computers! Navigate your way through a field of logic gates as you attempt to find the correct inputs to complete the level in this unique and fascinating puzzler.

**Objective**

The players must complete certain levels in order to complete the game. In each level a puzzle containing a circuit appears. In these levels they have to turn the correct switches on in order to guide the signal to the panel that says level complete and turn it on.

**Overall programming level**
Low - Medium. The game explains the circuits that contain conditionals to let the signals go through.

**Gladiabots**


**Description**

Assemble your robot squad, construct their AI and send them in the battle arena to see your strategy play out. Iterate, improve and repeat until you outsmart all your opponents and friends online.

**Objective**

The players must program the behaviour of their robot to combat with other robots. For doing so they have access to a behaviour tree that lets them program the robots to do things in certain conditions.

**Overall programming level**

Medium – High. The players have access to conditionals to control the behaviour of their robots. This lets the players to create behavioural trees that will execute actions in a loop if the conditional is fulfilled.

**Coding planets**


**Description**

Coding Planets is Educational, Brain training and puzzle game. In this game, you have to give instructions for a robot to solve the puzzles. Everyone can play the game.

**Objective**

The players have access to several worlds that contain several levels. For these levels to be completed you have to write the main program that will make the robot move and get all the gems.

**Overall programming level**

Medium. In this game the players have access to a main function that will be the one that runs the robot and other functions that can be run inside the main. For programming these functions, they have blocks that will represent certain actions. Each function has a limited amount of spaces so you will have to manage your resources in order to complete the levels.

**Tynker – Learn to code**


**Description**
Tynker is the easiest way for children to learn programming. Solve puzzles to learn concepts, easily build your own games, and control robots and drones using our new step-by-step coding tutorials. 50+ free starter templates included for games, apps, drawing, music, physics, and more. Additional puzzle levels available as in-app purchases.

**Objective**

The game is based around different puzzles. For completing each of these puzzles, the players must program the behaviour of the main character by combining certain blocks that represent various actions.

**Overall programming level**

High. For programming the behaviour of the main character, they have access to blocks representing certain actions, loops to repeat some of these actions as many times as wanted, conditionals and variable management.

**Coddy: World on algorithm Free**


**Description**

Challenge your logic in 120 unique levels in which you must help Coddy the bot, to collect all stars and reach the exit! You will help Coddy, and he will help you not only understand the basics of programming, but also to train your wits.

**Objective**

The game provides several levels for the player to complete. For completing these levels, they must program the behaviour of the main character. In order to do so, they have a main function that will execute all the blocks that represent certain actions inside them in the order they are placed.

**Overall programming level**

Medium - High. For programming the behaviour of the robot, the players have access to functions, conditionals and it also gives the option to debug the program before compiling it.

**Algorithm City: Coding game for kids**


**Description**

The game is a fun and innovative coding game to teach children the basics of programming and algorithm. It teaches coding with fun games with the animals. The children gain the basic concepts of coding, like command sequencing, functions and loops, by guiding the character by collecting golds and solving levels.

**Objective**
The game provides several levels for the player to complete. For completing these levels, they must program the behaviour of the main character. In order to do so, they have a main function that will execute all the blocks that represent certain actions inside them in the order they are placed.

**Overall programming level**

Medium – High. Medium - High. For programming the behaviour of the animals, the players have access to functions and conditionals.

**Bit by Bit – Programming Game**


**Description**

Bit by Bit is a game to prepare kids for learning programming.

**Objective**

The game provides several levels for the player to complete. To do so they have access to certain actions in form of circles that they can place in the different fields that when the characters step on them will make them do certain actions. You control more than one character at a time in certain levels. For completing the levels, you have to guide each character to the field with their matching colour.

**Overall programming level**

Low – medium. The game claims to teach planning ahead, problem solving and analytical and logical thinking. Nevertheless, you have only access to actions represented in blocks that get placed in the fields for the players to behave in some way.

**Coding Planets 2**


**Description**

Coding Planets is Educational, Brain training and puzzle game. In this game, you have to write program for a robot to solve the puzzles. There are three Programming languages supported and everyone can play.

**Objective**

The players have access to several worlds that contain several levels. For these levels to be completed you must write the main program that will make the robot move and get all the gems.

**Overall programming level**

Medium - High. Inside the game you have access to blocks that represent parts of the code that will make the robot do some things. The syntax is like the different languages that are
available in the game. You have access to loops in later levels, but no to conditionals or functions (it says not available).

**SpriteBox: Code Hour**


**Description**

SpriteBox is a unique puzzle-platformer; a mix of exploration and learning to code. By giving Sprite programs of instructions to follow, you can advance through unique worlds and help free Sprite’s bottled-up friends. During your adventure, you’ll learn to sequence commands, change parameters, debug faulty logic, and use simple to complex loops to solve problems. Once comfortable with Icon-based coding, you can choose to play SpriteBox using real Java syntax.

**Objective**

The game is a platformer where the players can move the player to complete the levels. In some parts of the levels they have puzzles that contain programming parts. These parts consist in programming the behaviour of a robot to place the needed blocks for the players.

**Overall programming level**

Medium - High. In the programming parts the players have access to loops and actions. You can change to java mode to make it look like real code.

**Coding game for kids – Learn to Code with Play**


**Description**

Kidlo Coding for Kids is a fun coding game to teach kids the very basics of programming, a very essential skill in today’s world. It teaches coding with fun games like firefighting and playing a dentist. Kidlo Coding helps kids improve their problem-solving skills, boosts memory and increases logical thinking skills.

**Objective**

The game provides several puzzles with challenges to complete. For completing these levels, they must program the behaviour of the different characters. The task to complete will be different in each level. For programming the behaviour, they have access to different blocks that will make the characters do certain actions like moving or interacting with things.

**Overall programming level**

Medium - High. The levels give access to loops, conditionals, arrays, debugging and some advance concepts.

**Human resource machine**
Description

Program little office workers to solve puzzles. Be a good employee! The machines are coming... for your job. From the creators of World of Goo and Little Inferno.

Objective

The players must solve different puzzles by programming the behaviour of the different workers they have access to. For programming the workers behaviour, they provide the players with different blocks of code that will interact with each other. Therefore, the players must organize these blocks in the correct order to make them do what they want and solve the various puzzles.

Overall programming level

Medium-High. The connection of the blocks is based in a procedural program language. The players must write procedural programs but using blocks instead of code.

Robotizen: Kid learn Coding Robot


Description

Robotizen is the #1 learn to code program for kids 4-9! Robotizen teaches the basics of computer programming and robotic through a variety of interactive learning activities in an interesting story. Robotizen would help kids to improve their problem solving, logical thinking and creativity.

Objective

The game provides several levels for the player to complete. In each of the levels they will have to program the behaviour of the main character. For doing so they have blocks that will represent certain actions. These will be placed in the main program that will make the robot act as told.

Overall programming level

Medium. The player has access to simple loops and actions for programming the behaviour of the robot.

Lobot - Robot Programming


Description
Lobot is a programming puzzle game. Give commands to Bot -the main character- to collect all Chips scattered around levels. Overcome the obstacles and avoid the enemies who ready to prevent Bot from completing the mission.

**Objective**

The game provides different puzzles for the players to solve. For solving these puzzles, they must program the behaviour of the main character. This program will make it move in a certain direction in order to get the chips that appear in each of the levels.

**Overall programming level**

Medium. For writing the programs the players have access to blocks that represent certain actions. These actions can be placed in the main function or in other functions. The main function is the one that will be played so in order to execute the other functions you have to place them inside the main function as many times as you want.

**Programming for Kids – Learn Coding**


**Description**

Board on to the magical adventure of coding, learning, and fun! Sharpen your logic and elevate your problem-solving skills with this brain-challenging games. Carefully designed with child-friendly puzzles structured in 3 basic programming categories which guarantee hours of mind-teasing entertainment.

**Objective**

The game provides several levels that contain different puzzles. In the puzzles the players must connect the dots like in the picture next to it. For doing so they have to program the movements of the pencil.

**Overall programming level**

Medium – High. For programming the behaviour of the pencil, the players have access to blocks that represent different actions. They also have access to functions and loops
Appendix C - Review of the games phase

Steam games

7 Billion humans

Description

7 billion humans is a puzzle based game where the players have to program the behaviour of the workers. When the players press the button to run their solution, each of the workers will execute the program at the same time. The objective of each level depends on the task that the manager tells you to do.

Design of the game

![Figure 3](7 billion humans' image 1)

The game is designed around levels that contain different puzzles or tasks. In each of these levels the players will be provided with different blocks that represent different actions the workers will perform when placed in the main program and pushed the play button. The actions will be run from top to bottom in the order that the players place them. The right part of the screen will represent the program where the players must place the correct action blocks in the correct order to complete the levels. In the left down part of the screen there is a button that represents the play button that will run the program. There are also buttons to run one step at a time and to stop the program from running, as well as one to reset the workers.

The whole left part of the screen is a representation of the puzzle with graphics. The workers will appear in an office like place with boxes containing a certain type of data. Normally the
tasks given by the manager will be to get certain boxes that appear in the levels and place them somewhere else or leave the room with them.

The progress in the game is represented by an elevator where each floor contains one of the levels. This elevator represents the progress of the players in the company. When you complete one of the levels you will get access to the next floor that will represent the next level the players will have to complete.

In each of the levels the players have access to hidden challenges for completing them. These will consist in developing the program using the less blocks possible and/or making their solution to be as fast as possible. Sometimes the solutions can be solved only completing one of the challenges at a time.

In the case of failing, the manager will talk to you complaining about not fulfilling the task. This dialog will be in red representing that you failed the puzzle. The other dialogs in the game appear in a white colour to differentiate between the ones meaning that you failed.

**Programming concepts represented in the game**

The game uses the different puzzles to make the players program the behaviour of the workers. For doing so, it gives them access to a main function that will run from top to bottom. In the main program they can place blocks that represent certain actions like moving or grabbing objects. It also gives them access to blocks that will simulate loops, for repeating certain actions as many times as wanted, and conditionals, to make the workers do certain things if a condition is fulfilled. The concept of debugging is also inside the game as there exists a button that lets them run the program step by step to look for errors in the program. With the blocks that represent actions there exist some that will let the players comment the program. Despite that these blocks only let the players to draw simple things instead of writing comments. In later levels, each of the workers have access to registers to store variables and do operations with them. This lets them change the values of the blocks that appear in the game and therefore change the variables that are being used.

**Else Heart.Break()**

**Description**

Else Heart.Break() is a game where you play as a boy named Sebastian. This character starts with receiving a job offer in another city about selling a soda for a company. When you arrive, you see that strange things are happening with computers and electronics in the city. After some time and by talking to the people living inside the city you can buy a device called modifier that lets you hack the different systems and modify their code for making different things. The players have complete freedom to do whatever they want, like talking to people, go to the café, sell soda etc.... In some cases, the players will be given certain tasks or information from other characters to continue with the plot of the game. The main task of the game that appears while playing is to find friends within the hacker and activist community that exists in the game to try to stop the people that are running the city.

**Design of the game**

The game is based in a point and click game. These games consist in clicking in the screen to move and interact with the different objects and characters available in the game. The
interactions with the objects depend on if what the players are clicking is an NPC or an interactable object. With the NPCs the players can talk to them and sometimes have a conversation to acquire information about the city or about them and their friends. With the interactable objects the players can sometimes use them, like for example the sodas or the coffees are drinkable, or safe them in their bag. There also exist other type of interactable objects that are static. These ones, like computers or lamps placed in the city, can be used or sometimes hacked. If used it is possible that some of them will do something relevant in the city, but when hacked another UI will appear for modifying the code of the object.

![Figure 4 Hacking UI in Else Heart.Break](image1)

This new UI will provide the players somewhere to write, compile or debug the code, undo their changes and some help available in the down part of the screen containing some useful functions or code to use.

![Figure 5 Else Heart.Break() conversation UI](image2)
The city is based in different zones with various objects for the players to interact. The other NPCs will walk through the city like if they are doing their own lives, so the players must learn their routine or common places where they can find them in order to speak with them. Some options will make them get mad at you or talk to you in a more friendly way, sometimes even inviting you to go with them to parties or places. The conversation with the NPCs is based in selecting the option you want (Figure 5) your player to say and the NPCs will answer in accordance.

![Image](image.png)

**Figure 6** Else Heart.Break() game

In the global UI the players have access to their bag where they can store a concrete number of items. By selecting one of the items the character will grab them and a small UI to interact with that concrete item will appear. Some items like keys will give you more options while interacting with other items, like in this case doors. Also the hacking system is divided into security levels, where at first the modifier (the item that lets you hack) will be of security level 0 letting the players hack only lamps and low security computers, but with a higher security level access they will be able to hack almost everything.

**Programming concepts represented in the game**

The programming side in this game is represented by hacking the different computers or hackable objects that appear through the game. When acquired the needed item for hacking the option “hack” will appear in computers like in figure 6. When using this option, the UI that appears in figure 5 will appear. In here the players will be able to make functions and program like if they were writing a real program.

The programming language used for coding is a pseudo code made by the developers called Sprak. [https://github.com/eriksvedang/SecretArcade/wiki/sprak](https://github.com/eriksvedang/SecretArcade/wiki/sprak). In this programming language the players have access to functions, conditionals, variables, arrays, loops and several other programming fundamentals. Several other programming concepts are
explained through the game, so players understand what they are, and they are not able to do.

**Human resource machine**

**Description**

Human resource machine is a puzzle-based game based around levels. In each of the levels the manager will give the players a certain task that the workers available in that level must do. For giving orders to their workers, they must write the program that will define their behaviour. This is done by placing different blocks that represent certain actions inside the main program table placed in the right side of the screen. Normally these tasks involve the grabbing and placing of blocks that contain data. The complexity of the tasks increases while going up in the levels. There is also some secret challenges and optional levels for the players in case they want to complete them. These challenges will appear later in the game.

**Design of the game**

The menu for selecting the different levels is represented by an elevator of an office, meaning that while you advance in the levels you are also acquiring some importance in the office you are working. Inside these levels, the players are given a task by the manager of that floor. In each of them they will have an entry zone, where blocks containing data like numbers or letters will appear, and an out zone, where they must deliver the different blocks. There is also a zone in the middle with which the workers can interact and place or grab blocks for solving the puzzles.

![Figure 7](image)

*Figure 7  Human resource machine level 9*

The UI in the game provides several buttons for running the program they made, running it step by step, increasing the speed at which the program plays, going back one step and stopping the current run. In the right side of the UI they have the blank part containing the program. In here they must place the blocks available just a little bit to the left side of it.
Each of these blocks represent an action the worker will do. In figure 7, the players have availability to the blocks for grabbing a block from the inbox and throwing it to the outbox (inbox and outbox blocks), copyfrom copies the block that the worker is grabbing and places it in the place desire of the middle space, copyto grabs a block from the middle space they specify, add adds the number of the block from the middle they specify into the block the worker is already holding, jump makes a jump to a position in the program and jump if zero jumps to a position in the program if the value of the block the worker is holding is zero. There are some others available through the game. For making the program, the players have also access for copying and pasting the current program into other blank spaces available for making alternatives that could solve the puzzle.

In each of the levels some secret challenges are given to the player to make their program as efficient as possible. These challenges are not unlocked until a later level as they are difficult to solve in general. When the players fail to solve the level, the manager will say something with the dialog bubble in red meaning that there is something wrong with the program.

The history of the game is based in the player fulfilling the tasks by giving orders to the workers and some dialogs with the managers as well as some cinematics containing several jokes.

**Programming concepts represented in the game**

In this game the programming is developed around making the program that will be executed in order to solve the puzzles. As explained in the design of the game there exist blocks that represent certain actions, some of them will be mathematical operations. Some of the blocks give access to making loops in the game as well as putting some comments.

For debugging the program, the game provides some buttons that will let the players go back one step in the program or run it step by step. There are also some challenges for making the program with the less lines possible and running with the making the less steps possible. This represents the need of efficiency needed in real programs while programming. There are also some kind of conditional while using the jumps as there exists some that will jump to the position of the program desired if the condition is fulfilled.

**Shenzhen I/O**

**Description**

In Shenzhen I/O you start working at a company that develops different circuits for various machines. It is based in a puzzle solving game where the players will be given tasks in each level they will have to complete. In this case the tasks are given via a message service, where your bosses and co-workers will tell them what they must do.

These puzzles are generally based around developing circuits for the company, but there are also some others that will require the players to play solitaire or other games. For developing the different circuits, they will have to use the necessary parts available in each of the levels and program them so the circuit can work. In the level, they will have access to a brief explanation of the task they have to fulfil as well as a verification tab where they check if the circuit is working correctly. Each of the components or parts they use have a cost, so the
The game has also some hidden challenges where the players are asked to develop the circuit as cheap as possible, as fast as possible and with the less lines possible.

**Design of the game**

For giving the players the tasks, they must fulfil in each of the levels, and at the same time developing a certain history, the game has a message system developed inside it. In this message system every, at the end of every level, the co-workers or the boss will write messages to the player telling them what their next task is or just simple company chatting (Figure 8).

![Shenzhen I/O message system](image)

**Figure 8**  Shenzhen I/O message system

In each of the levels they will have access to an UI containing fields where they can place different components for developing the circuits, a panel placed in the right side containing all the components available for that level, a panel placed in the down part providing the information about the task, the cost, the power usage, the lines of code, the verification of the circuit and several buttons for running the tests.
The game also comes with a manual explaining all the instructions needed for completing the levels, as well as examples of working solutions in case the players get stuck. In this manual, everything that needs to be known for understanding how the circuits being developed inside the game work is explained. Inside it there are also some jokes that will make it easily readable.

**Programming concepts represented in the game**

For solving the puzzles, the players must code the program of the circuits. For doing so they have access to different instructions, that like in an assembler programming language, they will control the different registers available in the components of the circuit for transmitting signals, values and other shorts of variables or data. Other instructions for making arithmetical operations, test instructions and basic instructions are also available to the players for solving some of the puzzles. Conditionals can be also developed by placing the “+” or “-” symbols in the code, which means that these lines will be enabled or disabled by test instructions. There is also the instruction jump that will let the players program loops for the program to run as many times as they want. The players can also comment the code for organizing the different parts of it and make it easy to understand.

In the manual of the game, there is a complete explanation of the different functions of the system as well as how these instructions work. There is also an explanation of the components available in the game for understanding how the program behaves in each of the components and what registers they have access to in them.

**While True: learn()**

**Description**

In the game you play as a man that has discovered that his cat is very intelligent and wants to know what he is saying. For doing so he starts making some research in pages to see what has already been done and how he can understand his cat. He founds some data processing forums that gives him tasks for processing different sets of data automatically. The data must
be filtered in order to arrive to its destination in the amount needed. Some of the levels will require to filter the data by its form, by its colour or some other specified parameters.

**Design of the game**

The game is based in different levels that will present certain tasks to fulfil. These tasks will be based around machine-learning and data processing. The players will be asked to make use of the different tools the game provides to filter the sets of data they give them. For doing so the tools they have access too are different blocks that filter data in different ways. Some of them will filter the data by its form, some by their colour, some will reduce the errors the filtering produces, and several other blocks exist inside the game. The programs can be tested as many times before running for the first time and complete the task. There is also a way to save the solutions the players make, so they use them un future levels. While running the programs the players can increase the speed at which they are playing so they do not have to wait a lot for the program to finish.

![Figure 10](image.png) While True: learn() self-driving car level

The history of the game is based on the main character searching for a way to find how to understand his cat. This will guide him to do certain tasks around data management and machine learning. As the project he is doing is expensive he needs to make money while learning, so for doing so the players can make companies that will raise money with time. Also, the different solutions can be run as many times as the players want in order to obtain money. They also need to consider that running costs money as they need to rent different servers to run the programs. With the money they earn they can buy different cosmetics for their character or some upgrades to their computer as shown in figure 11, so programs run more efficiently.
Programming concepts represented in the game

The game is based around machine learning and data processing. All the blocks that represent programming are made around conditionals and algorithms that sort the different sets of data the player is given.

There is also the concept of efficiency of the different ways to sort the data, as some of them will last more doing a similar task than others due to the complexity of the filter. Facts like filtering by the colour or filtering by the colour and the shape normally makes the program go slower, simulated by the different steps they must take for filtering something twice in the same program.

There is also the recursion inside the game as the players can make different solutions that can be reused in future levels or in the same level as many times as needed. With every level, there is an explanation of the different tools and how they work in their game but at the same time an explanation of what is the equivalent in the real world is explained (Figure 12).
While True: learn() explanation

Marvellous Inc.

Description

Marvellous Inc. is a game where the players must solve different puzzles to complete the levels. These puzzles are presented as tasks that are given by emails from a company, but sometimes these tasks will come from other co-workers or users of the program. These tasks will normally require the player to program the behaviour of the robot to complete it. In some of them, the robot will have to move and use objects, go to consoles to do mathematical operations with numbers or whatever the task requires them to do.

Design of the game

The game is based in different tasks that are presented by an email system (Figure 13). In this email system, the players will receive messages from the company or the co-workers asking them to fulfil certain tasks. This email system will act as a way to tell the “history” of the game, that is basically that you are working for a company.
The tasks mentioned above will consist in making the AI of a robot for solving a puzzle. First, for accepting the task they will have to press reply in the email, so then a tab in the UI with the word “code” will appear. In this tab as shown in Figure 14, the players will be able to write the code of the program that the robot will execute. This panel is the representation of the main function that will be executed line by line. The players also have access to several registers that appear in the down-left part of the screen, above the buttons that lets them run the program in different speeds as well as stopping or running it step by step.

The registers available to the players will let them write and make operations with different numbers, using instructions like add, mov or several others. In some levels, consoles will
appear in them where some of them will have numbers with which the player will have to play to fulfill the tasks. In other levels some objects will appear with which the players will be able to interact and do certain things.

![Figure 15 Marvellous Inc. manual and info tabs](image)

Inside the game there also exist two other tabs, the manual and the info tabs (Figure 15). Inside these tabs the players will have access to information regarding the different instructions available to use and information about the current task they must complete. They have also access to a button to give up in case the task was too hard for them to complete it.

**Programming concepts represented in the game**

In Marvellous Inc. the players must code the AI of the robot for doing certain tasks. For writing these programs they have access to several instructions for moving, interacting with objects, making mathematical operations, controlling the registers available in the game and making conditionals as well as loops. There is also a system for debugging the code where using some shortcuts will create breakpoints in the code, so it does not do any more steps than that line until the players tell it to.

---

### Exapunks

**Description**
Exapunks is a puzzle game where the players will have to solve them to complete the game. These puzzles consist in completing several hacking tasks by programming the behaviour of some little robots. These tasks consist in accessing some systems in order to get the files that are inside them, sometimes modify them, drop them in the output box or delete them and then disappearing without leaving a trace. These tasks are done due to the disease the main character has and so he needs money for getting the medicine needed that is quite expensive.

**Design of the game**

![Figure 16 Exapunks level interface](image)

In Exapunks for accessing the different levels of the game the players have access to the UI shown in Figure 16. In this UI the levels will appear after solving the last level available. The start of these puzzles will sometimes be by the AI that appears in the top-right side of the screen talking and making you several questions or telling you several important things to know where the history is going. In the down-right side of the screen, a chat with several other hackers talking will appear containing several funny messages or conversations about what is happening. In the left side is where the levels will appear and where the data of the completed ones will be shown. This data will tell you in how many cycles, what was the size of your program and the activity that was needed to solve that puzzle. This data gets compared to other players data to let you see how much you can improve your program. The levels can also be repeated as many times as wanted, to improve your score or to see what you did the last time you solve it.
Inside each of the levels, the players will have access to the screen shown in figure 17. In this screen, they will have to program the actions of the robots showing in the center of the screen for completing the task given, that appears in the bottom middle part of the screen, with the objectives in the bottom right. In the panel of the task the players have access to different buttons to run the program in various speeds, pause it or stop it as well as running it step by step for searching for errors. In the right side they have access to a button that will show the expected result in case they do not understand the objectives.

In the left side of the screen is where the different files available in the level appear, with information about the data they contain. In the top left side of the screen is where the robots with their name and program appear. Here is where the players can write the program of the robots to do as they want. Each of the robots has access to two register to store one piece of data each and can grab only one file. The F register will represent the data of the file the robot is currently pointing at and the M register will be for transferring files within the other robots.

**Programming concepts represented in the game**

In exapunks they must program the behaviour of the different robots to do as they want. For doing so they have access to different instructions that are like an assembler like language. These instructions will let the robots link to other spaces of the system, copy variables into their registers, making mathematical operations with variables and some other actions relevant inside the game. They can also make loops with the instruction jump in case they need as well as conditionals for jumping in the code too. They have also the concept of optimizing the program due to the secret challenges they have for making them in the minimum amount of cycles, lines and actions needed.
Robots: Create AI

Description

Robots: Create AI is a puzzle-based game where the players have to program the artificial intelligence of a robot to solve the different puzzles. In each of these puzzles they will have to write a program to solve that specific situation. The game does not have any history and the only objective is to solve these puzzles.

Design of the game

![Figure 18 Robots: Create AI level example](image)

Robots: Create AI is a puzzle-based game. In each of the levels the players will have to solve the puzzle by writing the program that will act as the artificial intelligence of the robot. For doing so, as shown in figure 18, the players have the left side part of the screen for writing the program. For doing so they can click to add the grey blocks that when clicking several options of what the players want the robot to do in that step appear. In general, for solving the puzzles the only thing the robot or robots must do is move to get to the flag.

In some of the levels, some blocks will disappear while running the program, so the players must create the program to solve the puzzle in every situation. For checking if their solution is working in every instance, the game will run the program several times and when all of them pass it will let the players go to the next level.

The players have access to an option called code log. This code log will show what movements and decisions the robot has taken while running the program. This is useful for finding errors that made the program fail.
Programming concepts represented in the game

For making the program that will simulate the AI of the robot or robots that appear in the levels, the players have access to different blocks that represent certain actions. Some of these actions will consist in moving but they also have access to conditionals, that will let the robot decide whether they should go one way or another and loops that will repeat certain actions indefinitely or until the player tells it to. In this game the effectivity of the algorithm they make is evaluated, as in most of the levels the program will be run more than one time to check if it is working for all the possibilities. The game also gives access to the log of the code seeing what decisions and movements the robot has made the last time the program was run to check for errors.

Spacechem

Description

Spacechem is a puzzle-based game where the players must make different systems for making the output of chemicals asked by the input of chemicals they have. For doing so they have access to two circuits, one of colour blue and other one of colour red. These two lines will be run concurrently, so the players must be careful to not collide the chemicals while they are running.

Design of the game

In Spacechem the players have access to a map of different worlds that will contain the levels (figure 20). In these levels they will have to create different factories that treat with chemicals, in order to output the chemicals needed with the inputs they provide.
In each of these levels the players will have to create a circuit for treating with the chemicals. They must consider that they have access to two lines that will play at the same time but there is not always the need of using them both. For solving the puzzle, they must create a circuit with these two lines for dragging and dropping the chemicals in the correct place and in the correct state. The inputs of the chemicals are made in the areas with that are made for that, these are the top left side for the red line and the bottom left side for the blue line. Sometimes some circles will appear in the board that will let the players unbind and bind the different chemicals to get the output needed. In later levels several factories will run at the same time.

![Figure 20 Spacechem level selection map](image)

![Figure 21 Spacechem level example](image)
Programming concepts represented in the game

In spacechem the programming concepts are not shown directly into the game, but programming thinking is involved in the process. At first, the players must consider that the lines the game provides run concurrently, that means that sometimes they will have to synchronize the lines for making certain things. In later levels several factories will be run at the same time so the synchronization between the inputs and the outputs of the different factories will need to be synchronized also. There is also some circles that represent certain actions and the lines represent main functions that play in a loop in the game.

TIS – 100

Description

TIS-100 is a puzzle-based game where the players must try to fix a broken unknown machine called TIS-100. For doing so they have to solve the different parts of the machine that are presented as puzzles. In these puzzles they can see how some segments are corrupted and cannot be fixed, but they can investigate the error it has by debugging them. For solving the puzzles, the players must write code in the segments that work in order to get the inputs to the outputs as asked in the task of that level.

Design of the game

Figure 22 TIS-100 level selection map

Spacechem is a puzzle-based game. In figure 22 shows the level selection screen where the different levels will represent one part of the machine that is broken. The main objective of the players is to fix this machine that no one knows how to fix. For doing so they must create a new program and solve the different tasks the game gives them to complete. In the UI of
selecting the levels different information about the best results of the players is available to see the efficiency of the programs they have made.

![Figure 23 TIS-100 level example](image)

In each of the levels a screen like the one in figure 23 will appear. In this screen the players can see the task they have to complete in the left top side of the screen. The left bottom part of the screen represents the different inputs that are expected to be received and the expected outputs the players are supposed to bring to the output points. In the example of figure 23 we can see how the players have access to the inputs IN.A and IN.B and the outputs OUT.P and OUT.N. The objective is to bring the expected outputs to the output fields by using the inputs as data sources that by doing the task they will let the players get the expected outputs they need.

**Programming concepts represented in the game**

In TIS-100 the players must develop the program of the different segments they have access to in the levels to bring the results to their respective outputs. For doing that they have access to instructions like those used in an assembler like programming language. Some of these instructions will let them move the data from one segment to another, do mathematical operations. In later levels they also have access to jmp instructions that let them jump to certain parts of the code creating loops or conditionals. There is another important mechanic inside the game and is the management of the variables inside the different registers they have access to in the segments.
Google Play games

Lightbot: Code Hour

Description

Lightbot is a puzzle-based game where the players must program a robot to complete them. In each of the levels, the robot will have to light the bulb in the blue colour fields. When all the blue field are lighted the puzzle will be completed, and they will get access to the next level.

Design of the game

![Figure 24 Lightbot world and level selection](image)

In Lightbot the players will have access to several levels that contain puzzles. At first, the players will be shown a screen for selecting the level they want to play. This screen is the one that appears in figure 24.

![Figure 25 Lightbot level example](image)
The puzzles available in the game are based on programming the behaviour of a robot that will have to light the blue fields that appear in each of the levels. As shown in figure 25, for making the program the players have access to different blocks that represent different actions in the game. These actions can be placed in the several functions they have access to but when the program is run it will only run the main function and what is inside it.

In the left side of the screen is where the representation of the different functions appears. Here is where they must place the different action blocks that appear in the down-middle part of the screen. These blocks can be placed by dragging and dropping them in the place they want them to be or by clicking them that in this case they will be placed at the end of the selected function. In the centre of the screen the visual representation of the puzzle appears. In the top right side of the screen the button for running the program appears, which execution can be stopped with the same button.

From the state of figure 25 the players should obtain the result shown in figure 26, with all the fields that previously were blue are now lighted in yellow.

![Figure 26 Lightbot level completed example](image)

**Programming concepts represented in the game**

In this game the players are required to make the program that the robot will run. For doing so they have access to blocks that represent certain actions and functions where they place these blocks to get executed in the order, they place them. With the tools they have available, the players can create loops by placing the execution of a function inside the same function. They must consider that the only function that will always get executed is the main function.

**Gladiabots**

**Description**

Gladiabots is a puzzle-based game where the players must make the tree behaviour of the artificial intelligence of several robots for them to do something. The objective in the levels is
to kill the robots of the enemy and acquire the different orbs that appear to score points. When the enemies are killed, and the points are scored the level will be completed and they will get access to the following levels. It also has a multiplayer mode to play against other players and show your skills for making the AI of your robots and your strategies. The robots have different weapons to give diversity to the game.

**Design of the game**

![Gladiabots main menu](image)

**Figure 27** Gladiabots main menu

Gladiabots is a puzzle-based game but it also has a multiplayer mode for playing against other players. In the main menu, the players have access to the different parts of the game. They can go and do the tutorial where the basics of the game will be explained, play the multiplayer mode, play the campaign of the game that consists in several levels with different puzzles to solve, sandbox mode that lets you try everything the game provides, team setup is for building your teams of robot for future use and access other outside resources of interest.

In figure 28, an image of an example of a level of the campaign is shown. In the levels of the game, the players will be shown with several robots being placed in the board. Some of these robots will be of blue color that will represent the player, and others of color red that will represent the enemy. In the board other items like the balls and the areas colored with red and blue will appear. The balls are the objects they must get and bring them to the areas of their color to score points in the game. In some of these levels they will also have a limit of time for completing them.
For completing the levels, the players must make the behavior tree of the different robots. For doing so they can access to the screen showed in figure 29. In this screen they will have a main block that will be the big white one that will represent the robot. The actions will be executed from the left side of the tree to the right side, so in case the first block is not possible or the condition it has to make that action is not fulfilled it will go to the nearest on in its right side. These will let the players configure the behavior of the robots in several situations that might occur. For making this behavior trees they will have to link them with each other. If they click and drag to make a cable and drop it in the middle of nothing the option of creating a new block will appear.

**Figure 28** Gladiabots level example

**Figure 29** Gladiabots behavior tree example

**Programming concepts represented in the game**
In gladiabots the players must program the artificial intelligence of the different robots they have access to in each of the levels. For doing so they have access to the screen showed in figure 29 where they make a behavioural tree. In this tree there are blocks that will represent certain actions that will be played if certain conditions are fulfilled. There exist some for attacking the nearest robot if it is at the middle or short range and if the condition is not fulfilled it will jump to the next action that could be moving in the direction of where the enemy robots are. This behaviour tree is played in a loop and checked every frame to see if an update of the situation of the robot has occurred. The players have also access to blocks that behave as conditionals, that when fulfilled it will play the action they connect to.

**Coddy: World on algorithm Free**

**Description**

Coddy is a puzzle-based game where the players must write the program of a robot to complete them. For doing so they have access to several blocks that represent certain actions that can be placed inside the main program, that when the button play is pressed, this program will run line by line making the robot execute the actions.

**Design of the game**

![Figure 30 Coddy level selection](image)

Coddy is based on a puzzle-solving game where the players are given certain levels that contain puzzles to solve. In figure 30, an image of the level selection scene is shown. In this scene, the players can select the levels they want to play, where those that appear with a star are the ones that have been already completed, and the ones with the lock are the ones that have not been unlocked yet. By pressing one of these levels, the players will be transported to the scene shown in figure 31.
As shown in figure 31, the levels are based in a UI containing the different blocks of code, panels with the functions and buttons that let the player run the program. In the right part of the screen the players have access to the different functions where they must place the blocks of actions. In the down part of the screen they have access to the blocks that represent the actions the robot will make, explained by graphics of the action it will make. And finally, in the left part of the screen they have access to the buttons that play and pause the execution of the program as well as access to block of configuration and level skipping.

The main objective in all the levels is to make the robot catch all the stars and get to the final field that is colored in blue.

Programming concepts represented in the game

In Coddy, the players have access to different blocks that represent certain actions. For making the main function they must place these blocks inside it or inside other functions that, then the functions should be placed inside the main function. For making loops, the players have access to a separate function called “for” that will be repeated a specified amount of times. The other functions that appear in the levels can be placed as many times as needed inside the main function. There is no way of making conditionals in the game.

**Algorithm City: Coding game for ids**

**Description**

Algorithm City is a puzzle-based game where the players must program a penguin to grab the different coins that appear in the levels. For doing so they have access to several blocks that represent certain actions in the game.

**Design of the game**

Algorithm city is a puzzle-based game where the players are provided with certain levels to complete. One of these levels is the one appearing in Figure 32. As shown in this image the players have access to a zone placed in the right part of the screen that represents the main
function that will be played when the button play is pressed, and several other functions that can be used inside the main one. For making the program that will run, in the down part of the screen the players have access to the different blocks that represent certain actions. These blocks, when placed inside the functions, will run step by step, making the penguin grab the different coins that appear in the levels. In the down part of the screen a button for running the program also appears. In the left part of the screen several buttons for going back to the level selection and the speed at which the program will be played appear.

![Algorithm city level example](image)

**Figure 32** Algorithm city level example

**Programming concepts represented in the game**

In algorithm city, the players are provided with different zones for creating functions. In these zones they can create a function that will call to itself making an infinite loop, that will run the same actions forever. However, these functions can also be placed in the main function as many times as needed. For creating these functions, the players have access to different blocks that represent certain actions in the game, that when placed in the main program, will run from left to right and top to down.

**Coding planets 2**

**Description**

Coding planets 2 is a puzzle-based game that contains several levels for the player to complete. In these levels, the players will have to make the program of a robot that need to get all the gems that appear in each of them. For doing so, they have access to blocks containing representation of different parts of code that when placed in the main function, they will run from top to down in order.

**Design of the game**
In coding planets 2 the players have access to different worlds that contain several levels for them to solve. As shown in figure 33, the players can select a world where the basic concepts of programming will be taught, but there exist another one that teaches the concept of loops. Inside these worlds, several levels will appear giving certain puzzles for the players to solve.
As shown in figure 34, the players will be provided with an interface where the top part of the screen will be the representation of the puzzle they have to solve, in the down part the different blocks that represent the parts of the code will appear and in the middle part the main function with the buttons to run the solution at different speeds will appear. For solving the different puzzles, the players have to place the blocks they are provided with in the correct way for the robot to obtain all the gems and complete the level.

**Programming concepts represented in the game**

In Coding planets 2, the players have access for making loops in the game that will repeat certain actions a specified number of times. The blocks used for making the program, also have a similar way of representing the code as in real programming environments.

**Spritebox: code Hour**

**Description**

Spritebox is a platformer game with some programming puzzles put in the game for continuing in some parts of the levels. The players will have to move the main character from left to right obtaining all the stars and saving all the little whale-shaped things for at the end getting to the flag and finishing the game. Throughout the levels several puzzles containing programmable parts will appear for the players to solve and continue with the level.

**Design of the game**

Spritebox is a platformer game with programming puzzles inside the levels. In each of the levels, the players will have to move the main character through the map obtaining the different stars and saving the friend that appear. In some places they will meet some special blocks that will start the programming puzzles.

![Figure 35 Spritebox programming puzzle](image)

As shown in figure 35, the main character will hit the special block with the head and the programming puzzle will start. These puzzles consist in placing or removing the blocks needed in the right places so the players can continue with their journey. For doing so they
have to place the blocks they have access to in the down part of the screen, in the main program placed in the right part of the screen. For running the program, they have access to a big play button that will run the program until finished.

When the program is finished, the players will go back to the state where they can move and jump as shown in figure 36, for continuing with the level.

**Programming concepts represented in the game**

In the several programming puzzles that appear throughout spritebox, the players have access for making different loops that will run certain actions as many times as wanted as well as to the actions itself presented as form of blocks that connect with each other. No form of making conditionals or other programming concepts.

**Coding game for kids – Learn to Code with Play**

**Description**

Coding game for kids is a puzzle-based game that contains several levels for the players to solve. These levels are separated in different categories where some of them will teach certain programming concepts. Inside these levels, the players will have to make certain tasks to complete them.

**Design of the game**

Coding game for kids is a puzzle-based game, where the players will be provided with levels to solve by making a program that will solve certain tasks. In the level selection, the levels are classified by the concepts they will teach and inside this groups several different games containing different tasks to make will appear. As shown in figure 37, several games will be provided for each category. Inside these games several levels will appear to the players where in each of the levels they will have access to blocks that represent certain actions as well as a main function where they have to place these blocks. For solving the task, the players have to make the program with the resources they are given and in the proper order.
Programming concepts represented in the game

In Coding game for kids, the players have access to making loops that will run a certain specified number of times, and functions that they can place inside the main function as many times as needed. There are no ways of making conditionals, but there is also a way for debugging the different levels for searching for errors.

Programming for kids – Learn coding

Description

Programming for kids is a puzzle-based game where the players have to make the program that will run for solving the puzzle that appears in each level. For doing so, they have access to different blocks that represent certain action inside the game. These blocks can be placed inside the main function or in other functions in the correct order, so when the main program runs the puzzle gets solved.

Design of the game
In programming for kids, the players will be provided with certain categories representing some programming concepts. As seen in figure 38, there exist 3 categories, functions, sequences and loops. Inside each of these categories, several levels will appear with puzzles inside each one of the for the players to solve.

![Level 1](image)

**Figure 39** Programming for kids level example

As shown in figure 39, the levels will present with a puzzle in form of a pencil needing to draw the same lines as in the left image. For doing so, they have access to blocks that represent certain actions in the down part of the screen, and a main function zone in the right side of the screen where they can place them in the order, they want them to execute. There is also a function zone that can be used if needed.

**Programming concepts represented in the game**

In programming for kids, the players have access to make loops that will run a specified number of times as well as functions that can be placed inside the main function to repeat the same lines of code as many times as needed. There is no option for making conditionals or to debug the program in the game.

**Human resource machine**

**Description**

Human resource machine is a puzzle-based game based around levels. In each of the levels the manager will give the players a certain task that the workers available in that level must do. For giving orders to their workers, they must write the program that will define their behaviour. This is done by placing different blocks that represent certain actions inside the main program table placed in the right side of the screen. Normally these tasks involve the grabbing and placing of blocks that contain data. The complexity of the tasks increases while going up in the levels. There is also some secret challenges and optional levels for the players in case they want to complete them. These challenges will appear later in the game.

**Design of the game**

The menu for selecting the different levels is represented by an elevator of an office, meaning that while you advance in the levels you are also acquiring some importance in the office you are working. Inside these levels, the players are given a task by the manager of that floor. In each of them they will have an entry zone, where blocks containing data like numbers or
letters will appear, and an out zone, where they must deliver the different blocks. There is also a zone in the middle with which the workers can interact and place or grab blocks for solving the puzzles.

![Figure 40 Human resource machine level 9](image)

The UI in the game provides several buttons for running the program they made, running it step by step, increasing the speed at which the program plays, going back one step and stopping the current run. In the right side of the UI they have the blank part containing the program. In here they must place the blocks available just a little bit to the left side of it. Each of these blocks represent an action the worker will do. In figure 31, the players have availability to the blocks for grabbing a block from the inbox and throwing it to the outbox (inbox and outbox blocks), copyfrom copies the block that the worker is grabbing and places it in the place desire of the middle space, copyto grabs a block from the middle space they specify, add adds the number of the block from the middle they specify into the block the worker is already holding, jump makes a jump to a position in the program and jump if zero jumps to a position in the program if the value of the block the worker is holding is zero. There are some others available through the game. For making the program, the players have also access for copying and pasting the current program into other blank spaces available for making alternatives that could solve the puzzle.

In each of the levels some secret challenges are given to the player to make their program as efficient as possible. These challenges are not unlocked until a later level as they are difficult to solve in general. When the players fail to solve the level, the manager will say something with the dialog bubble in red meaning that there is something wrong with the program.

The history of the game is based in the player fulfilling the tasks by giving orders to the workers and some dialogs with the managers as well as some cinematics containing several jokes.

**Programming concepts represented in the game**
In this game the programming is developed around making the program that will be executed in order to solve the puzzles. As explained in the design of the game there exist blocks that represent certain actions, some of them will be mathematical operations. Some of the blocks give access to making loops in the game as well as putting some comments.

For debugging the program, the game provides some buttons that will let the players go back one step in the program or run it step by step. There are also some challenges for making the program with the less lines possible and running with the making the less steps possible. This represents the need of efficiency needed in real programs while programming. There are also some kind of conditional while using the jumps as there exists some that will jump to the position of the program desired if the condition is fulfilled.

**Coding planets**

**Description**

Coding planets is a game where the player is provided with several blocks that combined in the main program will make the main character, in this case a little robot, move through the field made by squares. The objective of each level is acquiring every gem by developing a little program with the blocks they have access to in that level. When the program is finished the robot will stop moving and if by that time, he has obtained all the gems the player will have succeeded but if not, they will have to try again with a different approach.

**Design of the game**

The design of this game is based in the game genre knows as “puzzle games”. The players are given a set of tools they must use to solve the different puzzles that appear through the game. These tools available to the player are represented as blocks containing actions that will be played in the order you put them inside the main function and press the button play. The puzzles appear as individual levels where the main task is to make the robot get all the gems that appear in it.

The game is divided in 3 worlds where the first one will be an introductory one to the concepts of the game and to the simplest tools available. Each of the other worlds will introduce a totally new concept for solving new puzzles with new challenges.

The main character is a robot making the players think that what they are doing is programming the insides of that little robot that will do as they tell him to. The game by itself doesn’t have any history whatsoever so there is no other purpose than completing the levels that are available in the game.

**Programming concepts represented inside the game**

In this game for solving the different puzzles you are given a main function, representing the core function that executes when you run a program, where you must place different blocks that represent various actions. The existing blocks that represent these actions will play in the order you put them inside the main function, representing the order of execution that a real program follows.

In the first levels were the puzzles are simple you only have access to the main function but when you go to the other worlds you will get access to smaller functions that won’t play if you
don’t put them inside the main function as many times as you want them to run. The functions are represented the same way as the main function where you can place the different blocks inside them until you fill the spaces. After the set of blocks that a function will execute is defined you can place that function inside the main function so that when the turn to run the function arrives it will enter inside the function and play the actions that are inside it.

In the last world you get access to the loops that will repeat certain action until they are told. This makes a reference to the different loops available while programming where you can repeat the action that some lines of code do until you want them to.

The game also is based in retrying each level until you get the solution you want, letting you rerun each level as many times as you want and fix the errors you have inside it.
Appendix D - Classification of the games phase

In this appendix, the different tables related to the themes and the topics inside them that appear in the ACM Computer Science Curricula 2013 have been made. This was made for making the classification of the different games with the topics they represent.

<table>
<thead>
<tr>
<th>Table 4 Algorithms and design classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts and properties of algorithms</td>
</tr>
<tr>
<td>7 Billion humans</td>
</tr>
<tr>
<td>Else Heart.Break</td>
</tr>
<tr>
<td>Human resource machine</td>
</tr>
<tr>
<td>Shenzhen I/O</td>
</tr>
<tr>
<td>While True: learn()</td>
</tr>
<tr>
<td>Marvellous Inc.</td>
</tr>
<tr>
<td>Exapunks</td>
</tr>
<tr>
<td>Robots: create AI</td>
</tr>
<tr>
<td>Spacechem</td>
</tr>
<tr>
<td>TIS – 100</td>
</tr>
<tr>
<td>Lightbot: Code Hour</td>
</tr>
<tr>
<td>Gladiabots</td>
</tr>
<tr>
<td>Coddy: World on algorithm Free</td>
</tr>
<tr>
<td>Algorithm City: Coding game for kids</td>
</tr>
<tr>
<td>--------------------------------------</td>
</tr>
</tbody>
</table>

Table 5 Fundamental programming concepts classification

<table>
<thead>
<tr>
<th>Variable names</th>
<th>Basic syntax and semantic s</th>
<th>Variable s and primitiv e data types</th>
<th>Expression s and assignment s</th>
<th>Simple I/O includin g file I/O</th>
<th>Condition al and iterative control structures</th>
<th>Function s and paramete r passing</th>
<th>Recursio n</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Billion humans</td>
<td>X X X</td>
<td>X X X</td>
<td>X X</td>
<td>X X X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Else Heart.Brea k</td>
<td>X X</td>
<td>X X</td>
<td>X X</td>
<td>X X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Human resource machine</td>
<td>X X</td>
<td>X X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

LX
| Shenzhen I/O | X | X | X | X | X | X |
| While True: learn() |  | X | X | X | X | X |
| Marvelous Inc. | X | X | X | X | X | X |
| Exapunks | X | X | X | X | X | X |
| Robots: create AI |  |  |  |  | X |  |
| Spacechem |  | X | X | X | X | X |
| TIS – 100 | X | X | X | X | X | X |
| Lightbot: Code Hour |  | X | X | X |  |  |
| Gladiabots |  | X | X |  |  |  |
| Coddy: World on algorithm Free |  | X | X | X |  |  |
### Table 6  Fundamental data structures classification

<table>
<thead>
<tr>
<th>Algorithm City: Coding game for ids</th>
<th>Coding planets 2</th>
<th>Spritebox: code Hour</th>
<th>Coding game for kids – Learn to Code with Play</th>
<th>Human resource machine</th>
<th>Programing for kids – Learn coding</th>
<th>Coding planets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X X X</td>
</tr>
<tr>
<td>Arrays</td>
<td>Records/Structs</td>
<td>Strings and string processing</td>
<td>Abstract data types</td>
<td>Reference s and aliasing</td>
<td>Linked lists</td>
<td>Strategies for choosing data structures</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 6**: Fundamental data structures classification
<table>
<thead>
<tr>
<th>7 Billion humans</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Else Heart.Break</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Human resource machine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shenzhen I/O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>While True: learn()</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Marvelous Inc.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exapunks</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Robots: create AI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spacechem</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIS – 100</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lightbot: Code Hour</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game/Project</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gladiabots</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coddy: World on algorithm Free</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm City: Coding game for kids</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding planets 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spritebox: code Hour</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding game for kids – Learn to Code with Play</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human resource machine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming for kids – Learn coding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding planets</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7 Development methods classification

<table>
<thead>
<tr>
<th></th>
<th>Program comprehension</th>
<th>Program correctness</th>
<th>Simple refactoring</th>
<th>Modern programming environments</th>
<th>Debugging strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 Billion humans</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Else Heart.Break</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human resource machine</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Shenzhen I/O</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>While True: learn()</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Marvellous Inc.</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Exapunks</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Robots: create AI</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Spacechem</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIS – 100</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lightbot:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Hour</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gladiabots</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coddy:</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>World on</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>City: Coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>game for ids</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>planets 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spritebox:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>code Hour</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding game</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for kids –</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learn to</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code with</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Play</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>resource</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>machine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programing for kids – Learn coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding planets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>