Game Design Document (GDD)

Projet jeu sérieux - Ebucliper

Juan José Parra Díaz - Souvignet Nathan - Wang Xihao - Bonetti Timothée

Table of contents

1. Introduction

1.1 Links

Itch.io: https://crhonopose.itch.io/ebucilper
Github: https://github.com/Crhonopost/Projet-jeu-serieux

1.1. Scope of the document

This document is part of the course “Jeux Sérieux”, subject of the “IMAGINE” master's program at the University of Montpellier. It describes our prototype “ebucliper”, a serious game designed to teach the basics of programming, including fundamental concepts like instructions, variables, loops and functions. The goal of this project is to make a game that works as an introduction to programming for people that don't have a background in computer science and help them understand the most important concepts so that they are able to start programming with some fundamentals.

1.2. Elevator pitch

Ebucliper is a game in which you can learn to program while having fun. In the game, you can tell a robot to place blocks in a grid by creating a set of instructions. These instructions are orders that are sent to the robot and cover the fundamentals of programming, including variables, loops and functions.

2. Game Overview

2.1. Game concept

In this game you play as an engineer who is in charge of the reconstruction of the university. Your mission is to give instructions to a **Blocky 2000**, a construction robot, to rebuild the different buildings and elements of the university campus.

In the main gameplay, the player has 3 spaces on the screen:

- Construction zone: a 3D view of a grid in which you can find Blocky with the desired and the current structures.

- Instructions inventory: a group of available instructions that the player can use to create the structure.

- Coding area: a panel with the different chosen instructions that Blocky will follow.

The player has to “code” the instructions and send them to Blocky in order to make him do the necessary actions to build the target structure.

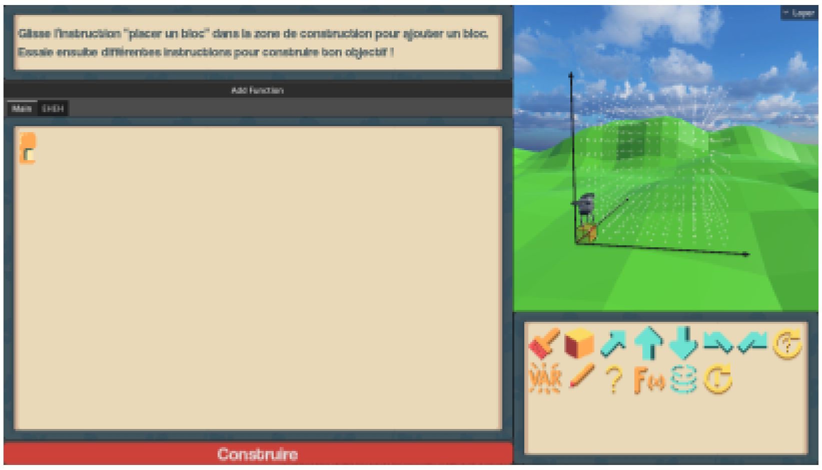

Glisse l'instruction "placer un bloc" dans la zone de construction pour ajouter un bloc.
Essaie ensuite différentes instructions pour construire ton objectif !

Figure 1 : View of the main scene

In the image we can see the main gameplay interface, with the different 3 panels.

2.2. Audience

This serious-game prototype targets players with little to no programming knowledge. Its goal is to introduce core programming concepts in a playful and challenging way, serving as a first step into the world of programming. It also allows players to discover more about the Campus Triolet as they rebuild iconic locations, such as Building 16, with the help of renowned researchers like Nicolas Lutz.

2.3. Genre

This game is a serious game. It can be seen as a puzzle game.

2.4. Setting

The game takes place at the university of Montpellier - Campus Triolet. The campus has been destroyed by the mutant plants of Tom's group. The professors of the Imagine Master ask for your help.

2.5. Core loop

You progress through levels earning stars by completing them. In each level you have to create a different structure using different techniques.

The player first selects a level from the world map. In each level he can read the mission's objective and some context of the level. Then, he uses the available commands and places them in the coding area by dragging and dropping them.

When the player is happy with the instructions he used he can send them to Blocky by clicking on the “Construct” button. This way, Blocky executes the code, moving and placing colored cubes.

If the final structure matches the target, the level is cleared and the player receives a star rating depending on the quality of the instructions he sent. The quality of the instructions is calculated by comparing the amount of instructions used with the optimal solution.

The player can choose between rebuilding the structure to get a higher rating or continuing to the next level.

The base learning loop of the game consists of 3 stages:

- Observe and think
- Execute and analyse
- Modify and retry

2.7. Look & Feel

Bright, slightly cartoonish science fiction industrial look inspired by university buildings. Pixel art’s interface with clean lighting and high readability.
Fresh tones, campus green, sky blue, light beige, instruction use distinct icons.
Emotional tone : relaxed, hands-on, and rewarding. Animated feedback and bubble sounds to reinforce player actions.

3. Gameplay

3.1. Objectives

The main objective is to rebuild the campus by completing all levels. A secondary objective is to collect as many stars as possible.

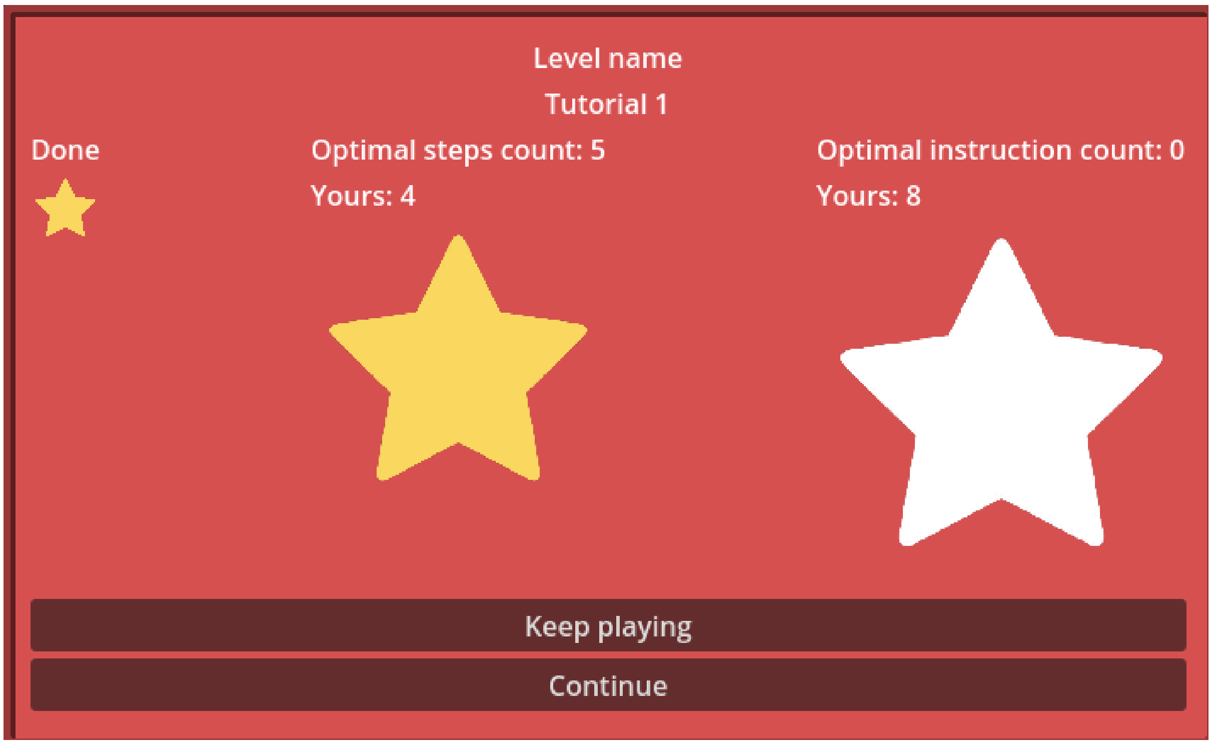
Stars introduce additional difficulty: the player is rewarded for finishing a level using the fewest instructions and the fewest execution cycles (each time an instruction runs at runtime counts as a cycle/step). Upon completing a level, the player earns between one and three stars: one for completion, one for beating the target cycle count, and one for beating the target instruction count.
But it is not implemented and the things we are looking at when completing a level are misleading.
One more day and it would have been done.

3.2. Progression

The player progresses through the game one level at a time, and must complete a level to access the next. Progression is gated by the total number of stars collected. This system provides an additional reward and allows the player to bypass a level that feels frustrating. Having more stars than the minimum required also demonstrates a solid understanding of the previously learned mechanics.

3.3. Difficulty

The difficulty will be insured by the complexity of the targeted structures. The more you advance into the game the more you have to use complex instructions / functions.



4. Mechanics

4.1. Rules

The player has to use the available instructions to build a structure that is identical to the target structure.

The player is limited by the set of actions we give to him, he can however arrange instructions the way he wants.

4.2. Game universe

Completing a level saves the player's performance and unlocks the next one. By completing a level, the game stores the number of stars obtained, best instruction count and custom code solutions.

The data system is implemented using Godot Resources for easy persistence across sessions.

4.3. Physics

Tween-based drone movement only. No gravity or collision detection needed.

4.5. Character Movement

The drone can face four cardinal directions, move forward and move vertically. The player can control the robot's movement by using instructions.

Movement and placement of the drone are interpolated smoothly by Tween animation.

4.6.1. Game menus

At the start of the game, the player can see a list of all the levels, from which he can choose among the ones that are already unlocked.

Once a level is chosen, the player is shown a screen containing an image and some dialogs that give him context about what the selected level is about.

When the player finishes all the dialogs, he gets into the building zone. There he can either return to the level selection or begin coding. He also has the option to display the target grid to understand the required outcome.

4.6.2. Saving

Once the player completes a level, the information about his performance is stored.

We use godot’s resources for data representation, it is easy to save the player's level progression.

4.7. Assets

Characters: Drone model with animation.

Environment: Voxel cubes with color materials, simple ground with texture and a sky with a skybox.

UI : Instruction icons, buttons, stars, images.

Audio : placement sound, success/failure jingles, calm background music.

5. Graphics and audio

5.1. Visual system

The game consists of a 3D environment containing a grid, in which the player can see the structure’s construction in progress. As the goal of the game is to learn programming while building a structure, the 3D grid is the main element that we need to render to make the game work.

We added a robot to indicate the player where the current building position is. This way he can understand how the construction process progresses and have visual direct feedback about his performance in the level.

We also added terrain and a sky so that the environment looks nicer, but the core of the game is in the grid and the structures.

5.1.1. Player camera

Given that we have one main 3D element in our scene, we only need one orbital camera.

Our camera is initially located in a position so that the whole grid is shown. The player can move the camera to have a better view of the structures he is building and see different perspectives of the desired structure.

5.1.2. Landscape

For creating a landscape we used two elements: a skybox and a terrain.

In our game, the landscape is not that relevant, it is only used for giving a visual enhancement on the construction zone.

Taking this into consideration, we decided to use a really basic terrain and skybox, so that the attention of the player is on the important elements, the grid and the structures.

5.2. Interface

Our interface consists of 2 sections: menus and gameplay.

- Menus: simple interface that shows the player the different levels and gives them context about the game and the levels they select.

- Gameplay: here the interface is used to show the player a list of the available instructions, the current instruction blocks and a short text that can be seen as a tip.

The interface is a key element of our gameplay. It allows the player to be aware of the current state of the level and to know how the robot will behave. Our game teaches how to code, so the interface works as a code editor: a key element in the programming process.

5.3. Audio system

The Audio is minimalistic just one music and a “boop” sound when a block is placed.

5.3.1. Game music

We used this royalty free music :
https://www.youtube.com/watch?v=FqI9cM6fczU&list=RDQMdz3pcLruhWU&index=8

5.3.2. Audio look & feel

We want to convey a chill vibe for the player to help him concentrate. The sound system must be relaxing and pleasing. That’s why the volume is kinda low.

6. Story and narrative

6.1. Backstory

The game takes place at the University of Montpellier, after some catastrophic events lead to the destruction of the campus.

6.2. Main plot

The player must help reconstruct the campus that has just been destroyed.

6.2.1. Plot progression

When the player completes levels, they help the campus get reconstructed. Each level, the characters will ask the player to rebuild a different part of the campus. In the end, the whole university will have been rebuilt.

6.3. Cutscenes

At the beginning of each level the player will be shown a small cut scene with some short dialogs. These will give the user the context he needs to understand what and why he is building.

7. Characters

7.1. Supporting characters

Our game has one main character: the robot Blocky 2000, which is controlled by the instructions the user executes.

There are more supporting characters which appear at the beginning of each level and give the user some context of the current level. These supporting characters include renowned researchers and university professors like Nicolas Lutz.

8. Game world

8.1. Levels

In our game, each level is a target structure. The name of the level shows which structure is going to be built. Each level has a higher difficulty than the previous one and can teach different programming concepts.

8.1.1. Tutorial levels

All of our levels could be considered as part of one big tutorial, however, we decided to consider our first levels as the tutorials. In the first level, the player will have to build a simple structure so that he can get familiar with the interface and the most basic programming concepts. Each level will add a new type of instruction that the user can use to build more complex structures.

8.1.2. Main levels

As discussed in the previous point, all levels are part of a big tutorial. The game itself is a tutorial. This causes that all our levels could be considered as main levels. As the player makes progress the levels can get more complex.

8.1.3. Optional levels

Some levels are unlocked by obtaining a certain amount of stars. The player could “skip” some of them, as long as they have enough stars to advance. However, if the player wants to complete the game, then they need to complete all the levels.
It is planned but not in the prototype.