Orenaut: Creation of an action/adventure videogame

Sergi Bons Fuses

Abstract— This article features the design, implementation and testing of an adventure/action video game in 3D, third person, with procedurally generated levels and simple graphic aspect, top-down view, implemented with C++ and Opengl called "Orenaut". The project will follow industry standard methods, such as the creation of a GDD (Game Design Document) which is a document where game characteristics are specified, such as game experience, gameplay design, models needed, etc. It is developed in OpenGL4.6 To develop a calm, mysterious, and challenging video game ubicated in a colorful animated world. The game is developed in OpenGL4.6, the models are designed with Blender, the textures are designed with Krita, and all the software used is either open source or freely available.

Index Terms— Action, Adventure, Game, Game design, Procedural generation, Test

1 Introduction

VIDEOGAMES have been gaining global presence in the recent years, and with the rise of new technologies, it has become easier than ever to create a good game, but these technologies that allow for fast production, by giving a higher layer of abstraction (such as Unity [1], Unreal Engine [2], or other Game engines) also abstract the lower layers of the design, hampering the customization, and also impose fees for the monetization of the product (even when maintaining free development tools).

Simultaneously, the gaming industry has risen in the last years to a big market, in which most of the cost of videogames production is directed to marketing, making some genres thrive (Mainly games as a service [3]) while others generally fall behind in priority (Single Player, One Time Purchase Games). This combination has led to some genres not benefiting from up-to-date technologies, that could lead to breakthroughs in the potential of such genres, as well as a potential interest from an enthusiastic target audience, that could put them up to par with the aforementioned types of games.

The action/adventure genre is one that has had many iterations, many of which have become of personal interest, and while few have tapped into procedural generation, most games in the genre have evolved into more complex mechanics, and most of them have earned their unique identity by mixing those elements with various other genres or sub-genres. For instance, action/adventure MMO's (massive multiplayer online), or dynamic platform jumping games, and this can be further investigated with an analysis of the state of the art as exposed below.

Our proposal aims to mix the action-adventure genre with the capabilities of procedural generation, allowing for high replayability, as well as benefiting from the genre specific focus that these (action-adventure games) have, allowing for a game that can be experienced uniquely by each player, and rewards exploration and level replaying, boosting the uniqueness of each interaction the player has with the game, while maintaining the freshness of the experience and the entertainment that comes with it.

2 Objectives and Methods

As a means to acquire a goal, that being the creation of an action-adventure videogame, some objectives and methods have been set and employed, as well as a state-of-the-art analysis.

2.1 State of the art

Through a series of analysis performed during the evolution of the project, the current state of the art surrounding the scope can be defined through a series of comparisons.

Currently, the most similar "active" game to the desired project result would be the one depicted in figure 1, "Spiral Knights" [4], a game that was published in 2011, and while game servers are still up, development is halted indefinitely and declared not of interest to the current developers [5]. This game implements a system aimed to replayability denominated "The clockworks", in which the world in the game is composed of a series of cogs, which depending on the hour you log into a mission, said mission will have one of the various layouts.

"Spiral Knights" is a cooperative action-adventure MMO, with simple 3D graphics, but pseudo-2D gameplay, in which the players (circled in blue in figure 1) cannot jump, and while there's height differences, they are only visual and hardly affect gameplay, since the playing field always remains in one height level at a time, and those don't overlap, making the difference almost merely visual, as can be seen in purple in figure 1. The game also implements a combat system based on weapons, as well as a variety of enemies (circled in red in figure 1).

Fig. 1. Screenshot of the game "Spiral knights" with various components circled

Another approximation to the action-adventure genre would be the one of "The legend of Zelda: The Wind Waker" and its successors "The Legend of Zelda: Phantom Hourglass" and "The Legend of Zelda: Spirit Tracks" [6]. This take takes more advantage of 3D, by making a set of 3D levels in which different heigh levels may interact (figure 2), but the world is static, and does not reward players for multiple iterations of the game.

Fig. 2. Screenshot of the game "The legend of Zelda: Phantom Hourglass" showcasing different height levels

An approach with more focus on height-axis playability can be seen in "Blue Fire" [7] (figure 3) a 2021 action-adventure game with a lower camera angle, and more aerial movement options for the player, more focus in movement and a more dynamic gameplay. This world also remains static and does not change throughout the game.

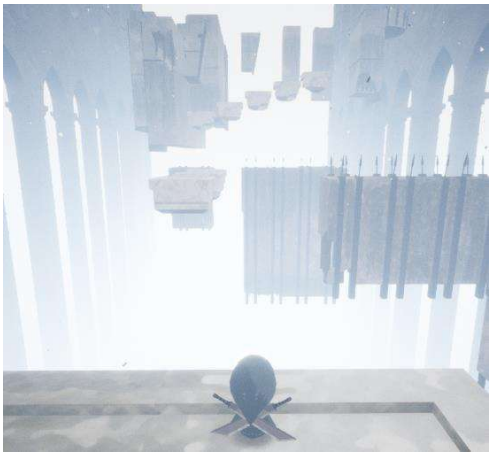

Fig. 3. Screenshot of the game "Blue Fire" showcasing map layout verticality

One last approach that defines the state of the art surrounding the project would be 2017, tequila works' game, "Rime" [8] (figure 4). In this specific adventure game, the story is told through interpretation and observing your surroundings, there is no dialogue, and all the information you need is delivered via cinematic, or world exposure, helping the game feel more mysterious, thrilling, and immersive.

Fig. 4. "Rime" promotional image, depicting it's artistic style

2.2 Objectives

The project main objective is the design, implementation, and testing of a videogame of the genre action-adventure which implements the following:

1. **Level procedural generation.** The level layout, and with them the level experience must vary every time the player decides to play one specific level, rewarding the player to play the same level more than one time, and avoiding repetitiveness as well as encouraging exploration of this same level multiple times.
2. **Simple combat system.** A combat system in which there's only one attack input allows easier multiplatform adaptation and more ease of mechanical balancing when considering enemy interactions.
3. **Player-Environment interaction.** The player will be able to interact with its environment to move forward, allowing for greater immersion, and

rewarding the player for taking unique approaches to each challenge, for instance throwing enemies into pits that are generated casually nearby.

4. **Graphics and 3D playability.** A tridimensional environment allows for greater freedom of action, both in game experience and design, transmitting a sense of freedom to the player
5. **Active Enemies.** Enemies that try actively to harm the player will boost its engagement, as well as reward its creative approach to every encounter, while rewarding high-skilled players
6. **Puzzles.** Mental stimulation is present, presenting players with problems which must be thought over to solve boosting player engagement.
7. **Simple animations.** To allow for low processing requirements, the game will have simple animation style, which requires low computational capabilities to process.
8. **Minimal HUD (heads-up display)**. To keep player immersion, HUD interface will be kept minimalistic, and will be highly integrated into the game.

2.3 Methodology

For the development of the project, a custom method has been used, one which follows TDD [9] Test Driven Development principles (manifesting requirements in form of tests, which must function for the result to be considered valid) mixed with agile methodologies. This is displayed in the project with the "TestMenu", a test case [10] selector that allows the developer to activate different tests, which are both Integration Tests [11] and Unit Tests [12]. This method allows for testing while the project is being developed, assuring the quality of the product before such is finished, allowing for real time adjustments in case project scope must be reduced, and allowing for dynamic testing. For design, version control and more, the development of this project has been done using a set of tools:

1. Blender [13]. Model and animation design tool, this tool serves the purpose of creating game models, as well as applying different to them.
2. Microsoft Visual Studio [14]. Integrated Development Environment (IDE) used for the implementation of gameplay, logic systems and C++ Programing
3. OpenGL [15]. Main graphics visualization library, necessary tool for visual tools and game graphics.
4. Krita [16]. Design tool used to draw the different required textures [17] for every model.
5. GLM [18]. OpenGL math libraries, used to help with data structures and math calculations.
6. GLFW [19]. OpenGL tool designed to handle window creation, keyboard interaction and more.
7. GLEW [20]. OpenGL extension wrangler, tool for graphics card handling, and OpenGL implementation.
8. SOIL2 [21]. Image processing library.
9. ImGUI [22]. Interactive GUI (Game User Interface) library designed for game development.
10. Git [23]. Version Control Software.
11. SFML [24]. Multi-media library, used for playing sounds.

2.4 Planification

For a quantifiable planification effort, a list of tasks was defined, as seen in table 1, as well as an estimated time of completion, to keep workload on track and adjust efforts properly.

TABLE 1
GENERAL PROJECT PLANIFICATION TABLE

Tasca	Description	Length Estimate (hours)
State of the art	Finding examples and surveying the current state of the art of the desired game	10
Documentation	Redaction of documents, and descriptive information about the project	30
Application Design	Design of the different parts involved in the application (models, textures, logic systems, etc.)	60
User Interface Design	Design of the display components.	10
Implementation	Implementation of the different designed components	190
Test	Intern and extern tests	100
Article and presentation	Documenting and perfecting this article and the following presentation of the project	40

3 GAME DESIGN

Design choices have been made, aiming at a cohesive, fun and interesting game. These choices determine the feelings the game transmits, as well as the experiences the players are pushed to feel. To accomplish this, the decisions concerning level, GUI, character and element design are summarized as follows, with the game desired experience.

3.1 Game Design Document (GDD)

For development purposes, by following industry standard, a GDD (Game Design Document) has been created. This document contains a detailed description of the game concept, systems, environment, scope, and gameplay. As well as original ideas for the game's GUI, Sound, and artistic design.
The main points of this document are:

1. **Game Concept:** in which the game idea is summarized

2. **Extended description:** in which the game is put into context, with state of the art comparisons, genre contextualization and idea exploration.
3. **Game systems:** in which the game desired systems are explained, such as board procedural generation, player control or interactive elements
4. **History/Narrative:** in which the game's overarching story is explained, as well as the narrative and its execution, and the characters designs.
5. **Art:** in which the game's art general theme is summarized
6. **Music/Sound effects**: in which the sourcing of music and sound effects is explained, as well as the genera idea surrounding its relevance.
7. **Technology:** in which the target processing requisites are explained.
8. **Demo scope:** in which the scope of the game's demonstration is determined.

3.2 Level design

General level design is aimed to be progressively more complex as the player advances. For instance, the first level expands only in one direction, making the exploration more linear. As the game advances, the levels expansion becomes less linear, and the importance of multidimensional thinking becomes more obvious when the vertical factor gains relevance, and puzzles start incorporating these elements.

3.3 Character and element design

Characters in the game follow two main themes: for the good/neutral characters, which only the main character forms part of in the current scope, the theme would be "static". Something that doesn't change, something that stands against time, such as stones, metal, or concrete. On the other hand, the enemies of such characters are based around "erosion". So, these characters draw inspiration from life forms, elements that represent change and that inevitably would crash and erode their opposites. This same erosion is also embodied in other elements that may harm the player, such as the flowing water in the first level, which supposes a passive threat.

3.4 GUI design

The game will communicate with the player minimally, and when doing so it will mimic the world environment, so it does not hinder immersion. For instance, in figure 5 we can see the player interacting with a stone tablet, (an element that will work as a reward for players that choose to explore the world around them). The image/text is dynamically displayed above the character's head, only taking away a part of the screen unlike more extensive GUI systems, and the world around the player is still on full display, and while the player attention may be shifted, the world around can keep moving, making an enhanced and more immersive experience.

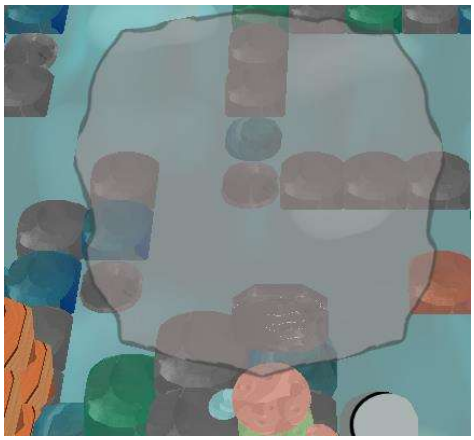

Fig. 5. GUI of player reading a stone tablet, with no text.

3.4 Game Experience

The game is meant to express a relaxed, tone, and while the end of the game works as a target to direct the player towards, the aim of the game is to hold player interest and capture its attention for as long as possible, throwing different procedurally generated "problems" that the player is free to resolve in creative ways. The music sets a happy but laid-back tone but paces up when more attention is required from the player, adjusting itself to trigger player reactions.

4 Game Development

The main result of the project is a functional game demo, which implements the objectives mentioned before. This demo will contain a set of models, textures, and gameplay logic. Those results are further disclosed in the following sub sections.

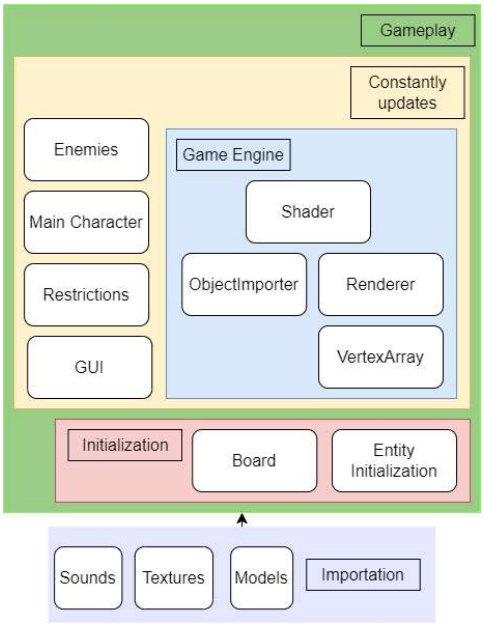

Fig. 6. Main game modules marked in different, with its main sub modules each.

4.1 Model Development

The models the game implements, along with their pertinent textures have been fully designed, modeled and textured from the ground up, using Blender and Krita, which is free open-source software so none of such may conflict with any copyright policy, and allowing further customization.
The list of models consists of:

9. 2 "*Tile*" models with 4 different textures (each)
10. 1 Enemy model with 1 texture
11. 1 Player model with 1 texture
12. 4 Interactable "*Tile*" models with 1 texture (each)
13. 1 Effect model

As well as some flat textures:

1. 5 Background textures, to make a cyclic background animation
2. 3 GUI readable textures.

4.2 Game Engine

Since the scope of the project starts with OpenGL, a tool is needed to develop a game, a tool which handles graphic visualization, and represents the interaction the player has with the game world (gameplay) in a way that the player can understand, this tool, or more precisely, this environment, is what is usually called a game engine [25].

This project's game engine has been developed to display models and apply different positional (translation, scaling, rotation...) and graphical (shading, transparencies...) transformations to them. It also has a (.obj) file importer, which allows models crafted in different software (blender in this case) to be drawn and modified by our own system. This game engine is comprised of a set of classes that abstract OpenGL code into visualization concepts:

1. ***"VertexArray"*:** A class designed to store visualization objects' vertices, is composed of other sub classes or structures:
   "*Vertex_Buffer*", class designed to store vertex values.
   "*VertexBufferLayout*", a struct that defines the inner layout for every vertex type.
   "*IndexBuffer*" A class that contains indexes for reusing vertices and swiften visualization pipeline.
2. ***"Shader"* [26]:** A class that processes shader files and prepares, which contain the main visualization program that is processed by the GPU [27], as well as allowing information transfer between CPU variables and GPU via uniforms [28]
3. **"Renderer":** A class that executes the visualization pipeline, binding and unbinding the appropriate OpenGL data whenever it's relevant towards visualization.
4. ***"ObjectImporter"* [29]:** A class that allows for model importing, allowing more complex models to be loaded, keeping the material, and texture they were imported with.

4.3 Gameplay

In a general scope, this system allows for the translation, rotation, scaling, and data processing of every entity, rendered or not, and does so using matrices following an MVP [30] system.

In a more detailed scope, this system initializes the data depending on level selection. When doing so, its preloads all models corresponding to the level in question, while generating the map layout for that instance. Aside from level initialization, the system provides responsive character actions, which range from entity actions (attacking, moving, losing health points, etc.) to board interactions (moving, breaking or modifying *tiles),* some of these interactions have repercussions that affect entity data (such as reducing character health) and are displayed to the player using a GUI system.
Entities are equipped with an in-built simple collision detector as well as a set of restrictions which are dynamically calculated and are hidden when not near enough the main character.
Those systems have its own specifications:

1. **Level procedural generation.** The game contains a system that defines "*Tiles*" which are the most basic metric unit in the game and are textured after rocks. These "*Tiles*" are grouped into 8x8xN groups called presets, in which N depends on the height of the level. Such presets present individual challenges, such as an enemy, a puzzle, hard to travel terrain, etc. Such challenges can be combined between various presets. The level procedural generation mixes a predefined set of presets using a "glue" (figure 7) technique, in which all presets are separated by a randomized padding that allows for unique interactions between presets

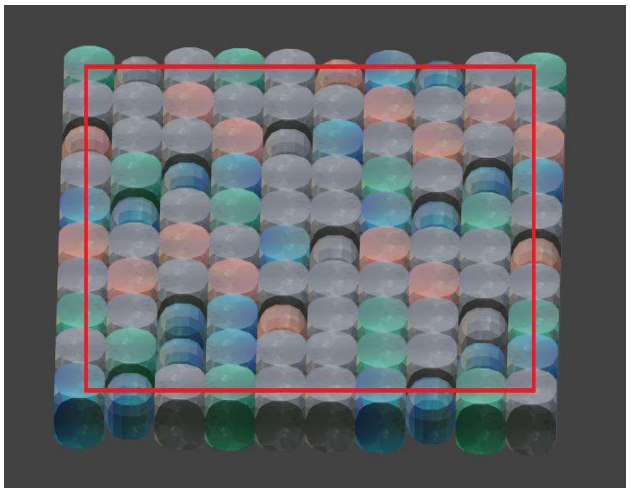

Fig. 7. Flat preset made of "tiles" with full "glue" marked in red

2. **Model occlusion.** To avoid excessive rendering and out of screen overload (by calculating things

that are not needed in the current frame) a limited display system is implemented, in this system, only the items close to the main character are loaded, with a customizable area of effect.

3. **Restriction system.** System that checks for player surroundings to allow or deny movement, as well as detect enemy impacts, works as a custom simple collision system.
4. **GUI system.** A GUI system in charge of displaying information to the player, this information can be either logistic information (such as health points, as depicted in figure 8) or informative (such as depicted in figure 5). This system melds with the environment, using models common to the game, to which the user ought to be accustomed to, to cause the minimal effect possible to the player's immersion.

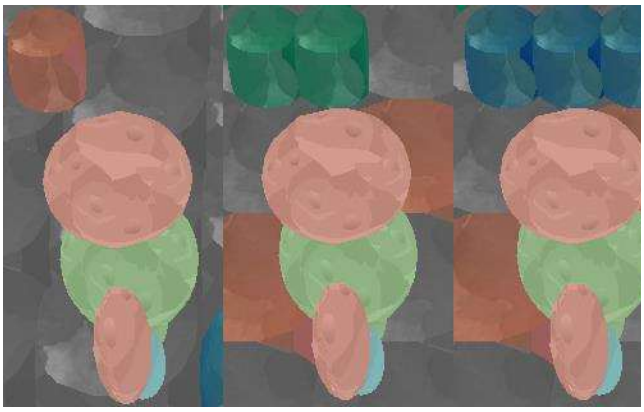

Fig. 8. Heath showcase in different health levels (1 to 3 health remaining from left to right)

5. **Combat system.** A system that allows interaction with enemies, player hp (health points), enemy hp and player/enemy position as displayed in figure 9.

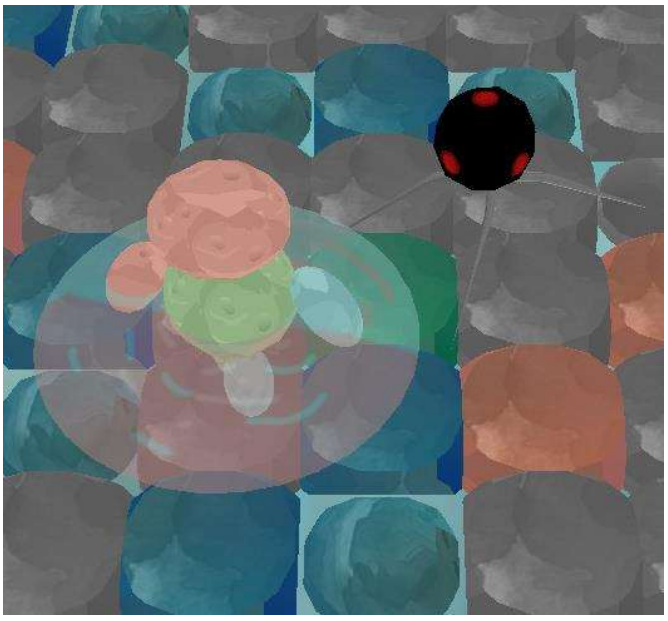

Fig. 9. Main Character Attacking an enemy entity.

6. **Entity Movement.** The player is allowed to move in all axis', using keyboard keys (WASD) for movement and (SPACEBOARD) jumping. Other entities move using an internal algorithm specific to that entity, that considers player position.
7. **Item Interaction.** The player can interact with a set type of *"Tiles"*. Some of these can be broken, some moved, and some can be read.
8. **Camera movement.** Most of the time, camera movement is tied with character movement, and follows a top-down perspective, angled at about -60 degrees to the horizontal axis, but under specific actions camera can be moved independently from the character to a certain extent.
9. **End.** The player interaction with the level ends once the player reaches the end tile, or once his hp amounts to 0.

5 Results and Testing

This project's main result is a game demonstration which implements the aforementioned concepts, as well as the independent modules of the project itself, such as the game engine, the textures and the music.
These results are backed by a set of tests, internal and external, which were implemented as requirements formalizations as in Test Driven Development methodologies.

5.1 Test

Tests have been present in the project since early development stages, so, a set of different tests have been developed. These tests manifest specific use cases [31], and there are unit tests [32] as well as integration tests [33] and performance tests [34]. These tests can be selected through an initial test menu (figure 10), and with ImGui different factors of how the application responds to these use cases can be visualized and contrasted.

Fig. 10. Test menu with different tests to be selected

From this specific set of examples some are unit tests (For

instance "Texture 2D", depicted in figure 11) which are meant to test a single capability of the application.

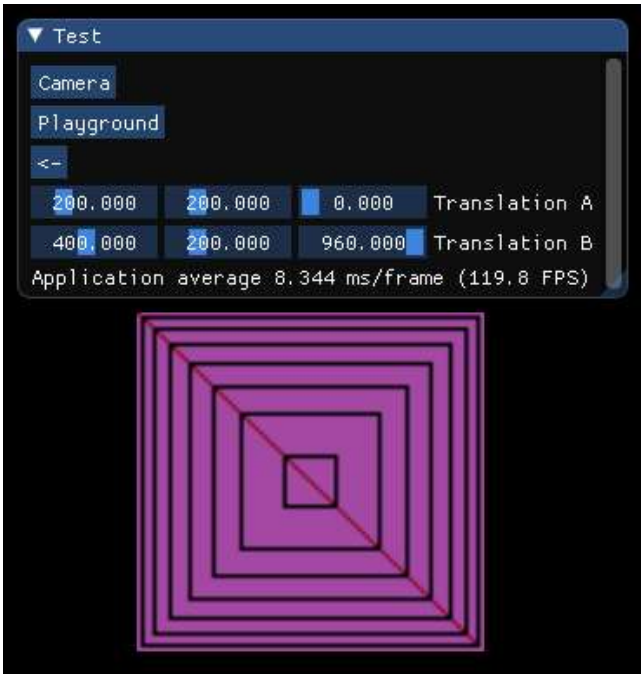

Fig. 11. Test Texture 2D which displays a single 2D texture of choice

Some are integration tests (for instance "MultiObject", depicted in figure 12), meant to test multiple capabilities of the application and the interactions between them.

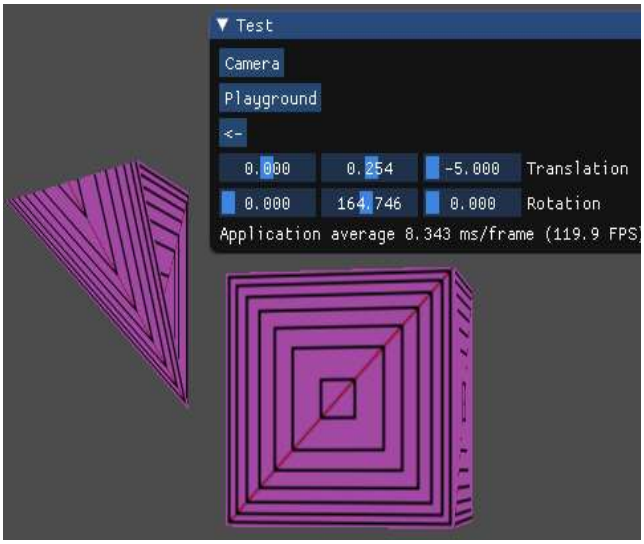

Fig. 12. Test MultiObject, which allows for multiple object primitives to be displayed, moved and rotated

Performance tests, (for instance "RainbowRoad") meant to gather information about the capabilities of the application as well as to contrast them, as seen in figure 13, in which the frames per second respond to the number of objects drawn, determined by the variable DrawColumns just over the frame per second (FPS) counter.

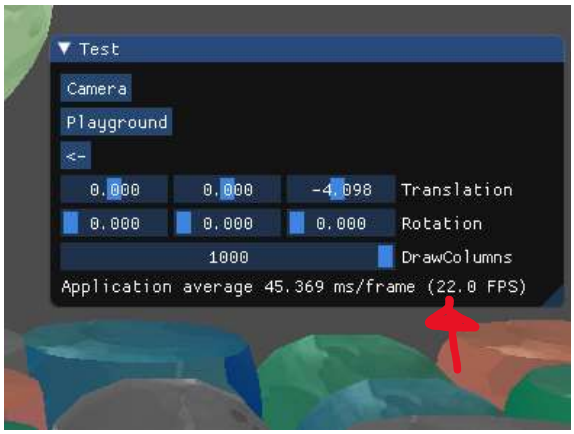

Fig. 13. Test RainbowRoad, which allows for a variable number of calls,
while displaying performance metrics

And a special test named "Playground" which serves as an exploratory test [35] environment, in figure 14 we can see the state of such test as of 10/06/2023 (in format dd/mm/yyyy) in which player movement was being tested, in an environment generated by other parts of the program.

Fig. 14. Test Playground, as of 10/06/2023, with systems such as board generation and player movement loaded

As for external testing, beta testing sessions have been performed as seen in figure 15, in which people of various backgrounds have tested the systems and experiences of the game, to perfect the systems and experiences intended.

Fig. 15. Beta testing for the full level test

6 CONCLUSIONS AND FUTURE WORK

Even when the project has been defined and scoped, its potential goes beyond a project of this magnitude, and as such the future work is worth evaluating with some closing thoughts.

As for concluding thoughts, all through the project, different problems have raised, many of technical nature, and many of design nature, since all models and textures have been drawn and made by one person, they cannot be considered of professional quality, but have been useful to develop different skills. The whole project has kept the maximum degree of modularity so all project components can be reused in future projects, and the project itself can be worked in for an upgraded final product that surpasses the scope of the current one.

While I had some experience in OpenGL, it was my first time creating a whole game engine from scratch, and although more work than initially thought, I think it has helped me develop my planification, technical, and creative skills, as well as my artistical ones.

As for the procedural generation, I think that it has been the part of the project that has generated the least number of problems, enhancing the initial thought pondered in the introduction of this same project, that many video game genres could be enhanced with the newest (and not so new) techniques, enabling many new kinds of games, that have not yet been explored, many of which have the potential of becoming really unique concepts.

6.1 Future work

As for future work, and complimentary things that this project will need to become a proper videogame:

- **Texture, Model, and lighting adjustments.** While the current models and textures work, a few visual updates would make the looks of the game more attractive, as well as help the overarching style gain more consistency, as well as some cinematic components.
- **Diversification.** While the current state displays the project's capabilities within a limited scope, diversification in entity type, as well as in board *presets,* would benefit a final product designed for a broader public.

6 ACKNOWLEDGMENT

As a closing to this project, I would like to express my sincere thanks to my tutor, Enric Marti Godia, for he has been pushing me to keep everything up to date as well as facilitating the proper material, guidance as well as the joy for the project.

I would also like to thank family, friends and my partner, for the support given throughout the project, and for tolerating me through my restless nights, days, and work-intensive time periods, and from these group, a special thank you to my younger cousin, who helped me with the music composition, as well as to my partner's little sister, who also helped in background animations.

Finally, one final thanks for all the beta testers that have been involved in the project, many of these overlap with the previous categories, but I still think that is a work that deserves acknowledgment on its own, and I truly think that without a single one of these people the project may have ended very differently.

References

[1] Unity technologies' Unity web page https://unity.com (Last visit as of June 2023)

[2] Epic Games' Unreal Engine web page https://www.unrealengine.com (Last visit as of June 2023)

[3] Bagga, Atul (October 10–13, 2011). Emerging Trends In Games-as-a-Service. Game Developers Conference. https://www.gdcvault.com/play/1015035/Emerging-Trends-In-Games-as (Last visit as of June 2023)

[4] Spiral Knights https://www.spiralknights.com Spiral Knights videogame web page. (Last visit as of March 2023)

[5] Gray Havens' game development declaration www.greyhavens.co. 2019. (Last visit as of March 2023)

[6] Wikipedia, The legend of Zelda Wikipedia page https://en.wikipedia.org/wiki/The_Legend_of_Zelda (Last visit as of March 2023)

[7] Fan-made Blue Fire's videogame wiki https://bluefire.fandom.com/wiki/Home (Last visit as of March 2023)

[8] Wikipedia, Rime (videogame) Wikipedia page https://en.wikipedia.org/wiki/Rime_(video_game) (Last visit as of March 2023)

[9] Wikipedia, Test Driven Development Wikipedia page https://en.wikipedia.org/wiki/Test-driven_development (Last visit as of May 2023)

[10] Wikipedia, Test Case Wikipedia page https://en.wikipedia.org/wiki/Test_case (Last visit as of May 2023)

[11] Wikipedia, Integration Testing Wikipedia page https://en.wikipedia.org/wiki/Integration_testing (Last visit as of May 2023)

[12] Wikipedia, Unit testing Wikipedia page https://en.wikipedia.org/wiki/Unit_testing (Last visit as of May 2023)

[13] Blender's Blender web page https://www.blender.org (Last visit as of January 2023)

[14] Microsoft's Microsoft Visual Studio web page https://visualstudio.microsoft.com (Last visit as of January 2023)

[15] Hewlett Packard Enterprise's OpenGL web page http://www.opengl.org (Last visit as of February 2023)

[16] Krita's design software web page https://krita.org/es (Last visit as of February 2023)

[17] LearnOpenGL, Texture tutorial web page https://learnopengl.com/Getting-started/Textures (Last visit as of June 2023)

[18] Hewlett Packard Enterprise's GLM subpage in OpenGL's web page https://www.opengl.org/sdk/libs/GLM (Last visit as of June 2023)
[19] GLFW, GLFW main web page https://www.glfw.org (Last visit as of January 2023)
[20] GLEW, GLEW main web page https://glew.sourceforge.net (Last visit as of January 2023)
[21] GitHub User SpartanJ's Soil2 GitHub repository https://github.com/SpartanJ/SOIL2 (Last visit as of January 2023)
[22] GitHub User ocornut's ImGui GitHub repository https://github.com/ocornut/imgui (Last visit as of June 2023)
[23] Software freedom conservancy's Git main web page https://git-scm.com/ (Last visit as of June 2023)
[24] Lauren Gomila's SFML web page https://www.sfml-dev.org/index.php (Last visit as of June 2023)
[25] Wikipedia, Game engine Wikipedia page https://en.wikipedia.org/wiki/Game_engine (Last visit as of June 2023)
[26] Wikipedia, Shader Wikipedia page https://en.wikipedia.org/wiki/Shader (Last visit as of June 2023)
[27] Wikipedia, Graphics Processing Unit Wikipedia page https://en.wikipedia.org/wiki/Graphics_processing_unit (Last visit as of June 2023)
[28] OpenGL Wiki contributors' Uniform (GLSL) web page http://www.khronos.org/opengl/wiki_opengl/index.php?title=Uniform_(GLSL)&oldid=14664 (Last visit as of June 2023)
[29] Object importer module supplied by *Visualització Gràfica Interactiva*'s professor in *Universitat Autonoma de Barcelona* and tutor of this project, Enric Marti Godia,
[30] Jordan Santell's article about Model, View, and Projection matrices https://jsantell.com/model-view-projection (Last visit as of June 2023)
[31] Wikipedia, Use case Wikipedia page https://en.wikipedia.org/wiki/Use_case (Last visit as of June 2023)
[32] Wikipedia, Unit testing Wikipedia page https://en.wikipedia.org/wiki/Unit_testing (Last visit as of June 2023)
[33] Wikipedia, Integration testing Wikipedia page https://en.wikipedia.org/wiki/Integration_testing (Last visit as of June 2023)
[34] Wikipedia, Software performance testing Wikipedia page https://en.wikipedia.org/wiki/Software_performance_testing (Last visit as of June 2023)
[35] Wikipedia, Exploratory testing Wikipedia page https://en.wikipedia.org/wiki/Exploratory_testing (Last visit as of June 2023)