



# **ASHESI UNIVERSITY COLLEGE**

## **BOARD-O-BACT: AN EDUCATIONAL SYNTHETIC BIOLOGY GAME APPLICATION**

**APPLIED PROJECT**

B.Sc. Computer Science

**Goodie Dawson**

**2021**

**ASHESI UNIVERSITY COLLEGE**

**BOARD-O-BACT: AN EDUCATIONAL BOARD-O-BACT  
SYNTHETIC BIOLOGY GAME APPLICATION**

**APPLIED PROJECT**

Applied Project submitted to the Department of Computer Science, Ashesi  
University College in partial fulfilment of the requirements for the award of  
Bachelor of Science degree in Computer Science.

**Goodie Dawson**

**2021**

## DECLARATION

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

G.B.A.D

Candidate's Name:

Goodie Blake Akorli Dawson

Date:

14/05/2021

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University.

Supervisor's Signature:

Elena Rosca

Supervisor's Name:

...Elena Rosca .....

Date:

...14/4/2021.....

## **Acknowledgements**

To my supervisor, whose encouragement and academic advice helped me undertake this project.

## **Abstract**

From small scale mobile apps to large scale esports, games have become one of the most pervasive creations across humankind in recent years. Although initially seen as simple pastimes, the cognitive value of games has increasingly become apparent. However, the use of games as educational technology is still relatively uncommon. Likewise, the Synthetic Biology field has been gaining a lot of traction. Applications in medicine, manufacturing and agriculture have been groundbreaking and yet only scratch the surface of what may be practically achieved. Unfortunately, interest in the field does not mirror the great value it holds. In this project, a desktop game application is implemented with the goal of allowing users to entertain themselves while casually learning about important foundational material in the Synthetic Biology sector as well as facilitating/refreshing their intrigue in the field.

## Table of Contents

Chapter 1: Introduction .....	1
1.1 Context .....	1
1.2 Personal Stake .....	2
1.3 Proposed Solution .....	2
1.4 Solution Significance .....	3
Chapter 2: Requirements.....	4
2.1 Requirements Elicitation.....	4
2.2 Procedure.....	4
2.3 Analysis of Survey Results .....	5
2.4 Key Insights from Interviews.....	6
2.5 Potential Use Cases .....	7
2.6 Functional Requirements.....	10
2.7 Non-Functional Requirements .....	10
Chapter 3: Architecture and Design.....	11
3.1 Architecture Overview .....	11
3.2 Development Model.....	12
3.3 Key Considerations .....	13
3.3.1 Physics System .....	13
3.3.2 Input Handling.....	13

3.3.3 User Interface .....	13
3.4 Key Modules and Control Flow .....	14
3.5 Low Fidelity Prototype.....	16
Chapter 4: Implementation .....	17
4.1 Technologies Used .....	17
4.1.1 Figma .....	17
4.1.2 Godot .....	17
4.2 Key Components .....	17
4.2.1 Nodes .....	17
4.2.2 Scenes .....	19
4.2.3 Resources .....	20
4.3 Key Implementation Techniques .....	20
4.3.1 Player Movement.....	20
4.3.2 Player Queue (Turn-based Gameplay) .....	22
4.3.3 Board Slot Actions.....	23
4.3.4 Save Structure .....	24
Chapter 5: Testing and Results .....	25
5.1 Component and System Testing.....	25
5.2 User Testing .....	25
Chapter 6: Conclusions and Recommendations .....	27

6.1 Requirements.....	27
6.2 Current Limitations .....	28
6.3 Future Development.....	28
6.3.1 Animation .....	28
6.3.2 Shaders.....	29
6.3.3 Game Logic Evolution.....	29
6.3.4 Tutorials.....	30
6.4 Final Conclusion .....	30
References .....	31
Appendix.....	32
A.1 Survey Results.....	32
A.2 Student Interview Guide Questions.....	36
A.3 User Testing Questions .....	37



# Chapter 1: Introduction

## 1.1 Context

From the creation of the discovery of new chemical and iron production methods and their facilitation of the industrial revolution in about 1760 to the day Thomas Edison patented the electric bulb in 1879 all the way to the 1960s when the internet's precursor ARPANET was born, human history is littered with many notable moments and fields of study that have drastically changed how we operate for the better. In the current world we live in, with so many promising fields, one may wonder, where can we next expect humanity's formative boom to emerge from? This capstone project hinges on the belief that micro/synthetic biology is one of the leading candidates for this next big change.

Microbiology refers to “the study of all living organisms that are too small to be visible with the naked eye” [1], whereas synthetic biology refers to the application of engineering principles to biology to allow us to redesign organisms in brand new ways in order for them to do some useful work. Although the field is new and exciting, public approval of its work seems mixed, largely due to most not feeling a personal stake in the advancements made [2]. This is quite intriguing as one does not have to do much research to discover many practical applications of micro/synthetic biology. In the agricultural sector (particularly relevant in the African context), synthetic biology is being used to do things like engineering insects for pest control; in medicine, it is being used to enhance vaccine development (a point that resonates even louder with the events of the ongoing pandemic), in manufacturing we now have self-replicating rubber and new green chemicals from agricultural waste[3], and these are only some of the uses we can find. When we

truly think about it, the possibilities of where we may likely go with these technologies is even more exciting than some of the extraordinary things being done now.

## **1.2 Personal Stake**

Consequently, it is here I find my personal stake in the project. In the second semester of my junior year in university, I found myself struggling to find a course that interested me enough to complete my course load. This was until the head of our computer science department informed us of a new synthetic biological engineering course we could take as a non-major elective. I was quickly hooked on the idea and immediately registered, yet when the idea was proposed to friends, they instantly turned it down. A bit of this stemmed from the aforementioned inability to see a personal stake, but a large part came from a strange preconception that the field was unreasonably difficult and, in the beginning, I almost started to think they might be right; the early housekeeping information we initially had to learn seemed quite daunting. Yet, by the end of the semester, I was more than glad I had chosen this path. The applications seemed so practical, and I almost felt I would be left behind in the world if I had not done this. So why were others so hesitant? Further conversations with my lecturer revealed this to be a problem that was not isolated to me, begging the question of how this strange bias could be addressed.

## **1.3 Proposed Solution**

The proposed solution for this comes in the form of a video game based on micro/synthetic biology. The game is inspired by a board game developed by the 2020 AshesiGhana iGEM team. The game in itself is a monopoly style game that allows players to gather the parts for and construct their own plasmid, key building components in the synthetic biology sector that everyone needs to

learn of. It then allows them to use these to degrade some arbitrary amount of plastic to help teach them the power and real-world applications these biological tools can have.

#### **1.4 Solution Significance**

The video game industry has grown over the years to become one of the most penetrative in existence, with the BBC reporting that the industry had become worth more than the video and music industry in the United Kingdom alone [4]. It is no wonder that as the industry grows in popularity, certain stigmas of video games (like “rotting” the brain) start to seem too rash, and the academic value of gaming in various sectors becomes more apparent. In this case, education is the area where games could facilitate massive gains. Studies have shown that when used as an educational tool, games have the capacity to boost a student’s learning in various fields, those of which include situational learning and knowledge integration [5]. What this means is that games have the capacity to teach students in a way that prepares them more towards applying the newly developed skills in a professional setting and allows them to consolidate what they learn with other knowledge models. This is all achieved while still providing the information in a fun and engaging package. What this means is that not only does this solution serve to address our goals of piquing new interest in the micro/synthetic biology sector and plateauing the relatively steep learning curve many struggle with, in the beginning, but it also does this in a way that encourages knowledge retention and pushes enthusiasts to use this knowledge for something purposeful. As this idea was generated from my personal experience as a university student, I find this point particularly meaningful as university students tend to be exposed to mountains of information during their tenure to the point where they go through extreme levels of abstraction and toss away much of what is learnt; this solution seeks to make sure that does not happen.

## **Chapter 2: Requirements**

### **2.1 Requirements Elicitation**

This portion of the project covers the process of gathering and analysing data in order to generate requirements to help contextualise the problem. This step is key in helping to streamline the development process and make sure there is a clear direction and guide that we can always fall back on to push the project forward. Seeing as this project seeks to flatten/plateau the steep learning curve that students often feel when they encounter synthetic biology and encourage interest in the field, three key users were identified: students, potential students, and lecturers. These were selected as it was felt that they gave a holistic view of the subject space and would provide the most relevant data for the planning of the project.

### **2.2 Procedure**

It was decided that interviews would be one of the chief tools in the primary research process as data was gathered to help generate and finalise the list of requirements that would describe the goals the project would seek to meet. This was targeted towards the student and lecturer group as the sample size for these group was relatively small. This presented the opportunity to give each individual more attention in the research rather than hand them a generic survey. Interviews in this way tend to provide richer information, allow for the extraction of themes across respondents and, due to the organic nature of the semi-structured interview approach used, allow for insights into points that may otherwise have not been touched in a simple survey.

Students and lecturer interviews were treated slightly differently, with student interviews focusing more on their experiences learning in the field and lecturer interviews focusing more on their experiences teaching in the field. The guide questions for the student interviews are provided

in the appendix under section A.2. Unfortunately, interviews with lecturers did not occur as planned.

Potential students, however, were queried using a simple online survey. This was done as their numbers were much greater, and the data needed from this group did not need to be as personalised as the former groups. The survey contained questions designed to assess student's perception of games as an educational and also assess their perception of Synthetic Biology as a field. This would help to prove the existence and relevance of the problem. The questions and results from this, in the form of pie charts and bar graphs, are provided in the appendix under section A.1.

### **2.3 Analysis of Survey Results**

The survey done focused on a total of about 70 young adults currently enrolled in university as this constitutes some of the most formative years of a person's life and would serve as a good case study into how/why people at this age may choose to pursue or not pursue certain academic tools or fields. The majority (98%) of the target demographic fell within the age range of 20-24 years old, with about 1% falling between the 15-19 years old category and 1% being older than 24. About 66% of these participants were female, with the other 34% being male.

Interestingly, the overwhelming majority, 94%, indicated they had used games as a learning tool before, and of this majority group, 88% said they enjoyed this experience, with the remaining 12% reporting a mixed experience; no one reported having a bad experience. More importantly, though, 88% of those who had used games as a learning tool prior again indicated that the experience was genuinely helpful, with 11% reporting a mixed experience and only 1% indicating that it was not helpful. Regardless, 88% of the entire participant group agreed that games

could be a good tool for introducing new fields of study, and a slightly lower 85% agreed games could be a good tool for furthering study in already familiar fields of study. Overall, it seems the data shows a very positive outlook on games as a learning technology by people.

On the other hand, participants showed much less awareness of Synthetic Biology, with only 30% of participants knowing about the field coming into the survey. A brief description of what Synthetic Biology is was then provided to allow those without prior knowledge a chance to form their initial impressions of the field. The responses to the next questions seemed to echo some information discovered during the secondary research process as although 59% of participants agreed to the practicality of the field, with a further 36% thinking it might have some practical value, only 17% of participants felt any personal stake in the field; 49% were indifferent, and 34% felt no personal stake whatsoever. Although, the final questions give some insight into this as 33% of participants gave the field the highest possible rank for potential difficulty and a further 40% did the same for potential steepness of the learning curve. University students tend to be incredibly stressed and busy already, and the idea that a course might be difficult combined with that of a steep learning curve is more than enough to put them off.

Clearly, Synthetic Biology does not have a good reputation amongst students, yet at the same time, games show great potential as tools to encourage more people to take an interest in the field.

## **2.4 Key Insights from Interviews**

The following are key themes discovered from interviews carried out with prior Synthetic Biology students. These students consist of two females and one male of all falling within the 20–24-year-old age range.

- The theory content introduced at the start of the course was the most difficult aspect of the course.
- Students are very open to using a game as a learning tool.
- Students would like a gaming tool to have some form of data persistence to allow them to revisit learnt content.

## 2.5 Potential Use Cases

The following section lists out a series of potential use cases for the solution based on the data gathered thus far. The tables introduce potential flows for how a user might interact with the proposed solution and outline some of the considerations that might need to be made.

Table 2.1: Use Case 1

Use Case Name	Lecturer looking to introduce course material to new students
Preconditions	N/A
Primary Actor	The Lecturer
Basic Path	<ul style="list-style-type: none"> <li>-The lecturer introduces the game to their class.</li> <li>-Students play the game during their free time.</li> <li>-Students come to class with a better understanding of some course content.</li> <li>- Lecturer fills in any gaps students may have.</li> </ul>
Alternative Path	<ul style="list-style-type: none"> <li>-The lecturer introduces the game to their class.</li> <li>-Students play the game during class.</li> <li>-Lecturer fills in any gaps students may have.</li> <li>-Lecturer introduces more advanced content which students can now grasp.</li> </ul>

Table 2.2: Use Case 2

Use Case Name	Student looking to learn some course material
Preconditions	Student already has prior knowledge in field* (For alternative path)
Primary Actor	The Student
Basic Path	<ul style="list-style-type: none"> <li>-Student begins Synthetic Biology class.</li> <li>-Student attends class and struggles to grasp early material.</li> <li>-Student plays the game during their free time.</li> <li>-Student returns to class with a better grasp of foundational material</li> </ul>
Alternative Path	N/A

Table 2.3: Use Case 3

Use Case Name	Student looking to casually play some games
Preconditions	N/A
Primary Actor	The Student
Basic Path	<ul style="list-style-type: none"> <li>-Student casually plays the game during their free time.</li> <li>-Students casually gains new knowledge from playing.</li> <li>-Students come to class with a better understanding of some course content.</li> </ul>
Alternative Path	<ul style="list-style-type: none"> <li>-Student casually plays the game during their free time.</li> <li>-Students gains no new knowledge from playing.</li> <li>-Students interest in the field is reinvigorated by the enjoyable experience.</li> <li>-Student continues to study the field.</li> </ul>



Table 2.4: Use Case 4

Use Case Name	Potential student looking to get a taste of Synthetic Biology
Preconditions	The potential student has no prior Synthetic Biology experience
Primary Actor	The Potential Student
Basic Path	<ul style="list-style-type: none"> <li>-Potential student discovers the game.</li> <li>-Potential student casually plays the game.</li> <li>-Potential student gains interest in Synthetic Biology.</li> <li>-Potential student enrolls in a Synthetic Biology course becoming a student.</li> </ul>
Alternative Path	<ul style="list-style-type: none"> <li>-Potential student discovers the game.</li> <li>-Potential student plays the game.</li> <li>-Potential student gains interest in Synthetic Biology.</li> <li>-Potential seeks out an expert (e.g. lecturer) to gain more information on the field.</li> </ul>

The use cases above cover the bases of the various types of users expected for the proposed solution. The first addresses lecturers and instructional personnel who may be interested in using the solution as a learning tool for their pupils, showing the different ways this could be implemented inside or outside the classroom. The next two focus on students themselves and the various contexts in which they could potentially gain from using the solution, taking both educational and conversational value into account. The final use case then covers potential students and paths of use that could lead them to pursue Synthetic Biology in some way outside the game.

## **2.6 Functional Requirements**

1. The user should be able to visualise the subject matter when playing the solution.
2. The solution should have a reference point where users can review material learned.
3. The solution should deliver approved synthetic biology subject material as they play.
4. The user should be able to save specific pieces of information they come across and consider important to their device.
5. The user should be rewarded for progressing through the solution and retaining information.

## **2.7 Non-Functional Requirements**

1. It should be easy for the user to learn how to play the solution.
2. The solution should limit any confusion to the user as they play.
3. The solution should have good replay value, encouraging users to return to it often.
4. The solution should be lightweight and not intensive to allow it to run well on users' different devices.
5. The user should feel excited to use the solution rather than it feeling like a chore.

## Chapter 3: Architecture and Design

### 3.1 Architecture Overview

The software architecture serves as a blueprint for the proposed solution and aids in guiding the development process. Since the proposed solution is a game, this section will look somewhat different from a conventional software architecture document but will seek to address the same compositional concerns. A pseudo-layered architecture has been chosen for the solution as it helps to abstract the solution into clear sections with their own components, and with the help of arrows, we can help to show how these interact with each other on a high level.

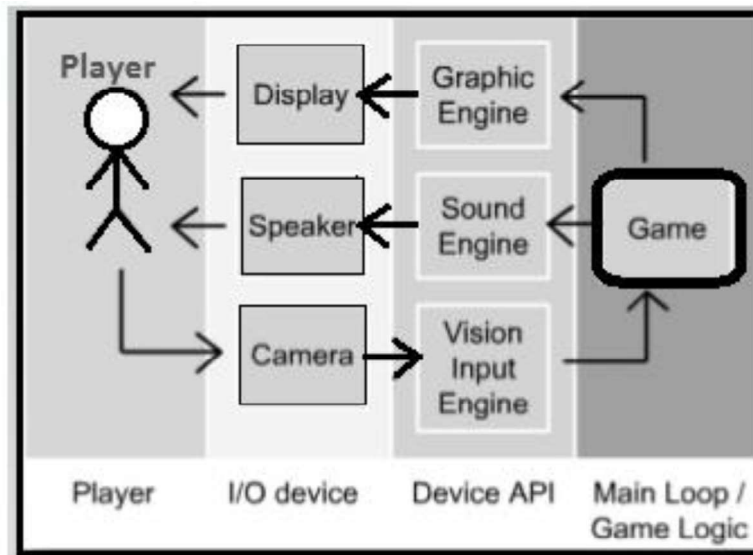


Figure 3.1: General game structure [6]

The above image gives an overview of the usually key components that come together to form a good game structure for the modern game. This served as a good starting point for architectural design by highlighting some considerations that might need to be made in designing the solution and deciding which development tools would be appropriate/necessary.

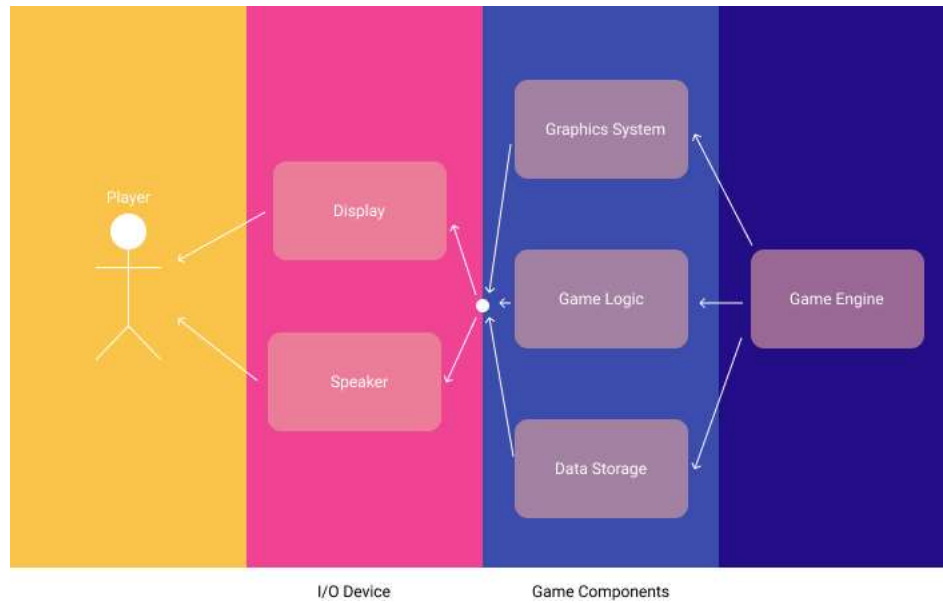


Figure 3.2: Abstracted structure inspired by figure 1

The above diagrams show how the development process has been abstracted so far from the initial diagram. The Godot game engine has been selected for the development of the solution based on this abstraction. It allows for the development of the game logic, storage and graphics integration all in one environment and is ideal for 2D and decent for 3D development. Game logic will be handled through various scripts and code written in the Godot GDScript languages, data storage via Godot resources and graphics implemented through Godot nodes. This will be further elaborated on in chapter 4. All these components come together to interact with the player through I/O devices, which in this solution's case are constrained to a simple display and perhaps speaker audio.

### 3.2 Development Model

In deciding which development approach the project would utilise, many features had to be considered. Some of the primary considerations include, but were not limited to, the scale of

the project, how testing would be carried, flexibility and the linearity (or lack thereof) of the project. In the end, it was decided that the agile methodology was most appropriate for this project and would be consequently used. This approach is heavily in line with the user-oriented nature of the solution as it places a heavy focus on customer involvement and satisfaction. It is more appropriate for the relatively small size of the project and allows for some malleability in requirements that could change with time and with testing, which can and will be conducted concurrently with development to provide a bespoke and relevant solution rather than a more general and passive one.

### **3.3 Key Considerations**

#### **3.3.1 Physics System**

This system handles how the game object interacts with each other. This is particularly important in properly determining collisions and various dynamics. Hence, when objects come into contact with each other, they should respond appropriately. This will be handled at the programming level using tools provided by the game engine. Godot uses a node structure that makes the development of this module quite intuitive.

#### **3.3.2 Input Handling**

This system handles the input the user puts into the game. If the user clicks on something or presses a key, the appropriate response should be triggered. This is a key part or result of the game logic and hence will be handled at a programming level.

#### **3.3.3 User Interface**

This is the interface that allows communication with the user in the game. This is important in conveying information to the user and allowing them to navigate the game properly. An

advantage of the Godot engine is that many of the tools used in scene building lend themselves to designing good, responsive user interfaces.

### 3.4 Key Modules and Control Flow

Before development could properly begin, it was necessary to develop a blueprint of the key events that would make up the application and outline how the flow of data and control may be structured between these key components.

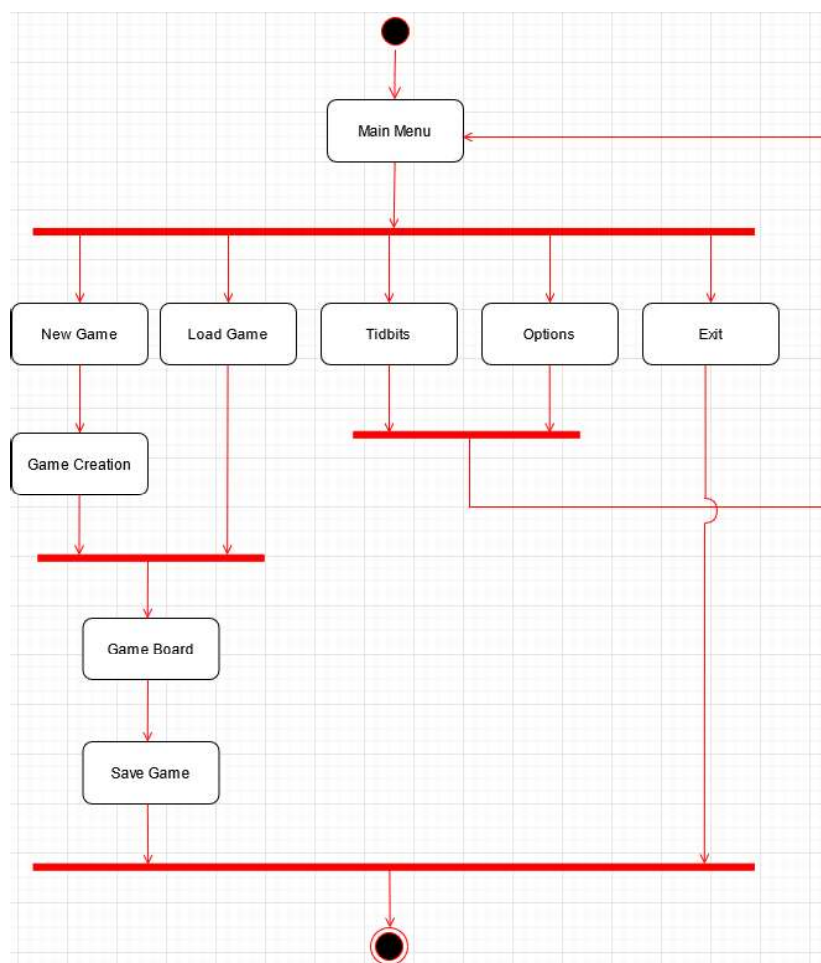


Figure 3.4: Activity Diagram

The preceding activity diagram shows the workflow of the application and how the users choice affects their navigation of the solution. Activity diagrams are often described as an advanced or enhanced version of a flowchart which is what makes them good tools for discussing the flow of control through the application. The initial and final states of the application are represented by the two black circles, the initial being on top and the final being at the bottom. The rounded rectangles represent activities. The arrows indicate the flow of control from one state to another, and the horizontal bars are forks/joins. Forks start with one arrow and split into multiple, whereas joins take multiple arrows and combine them into one.

As we can see, the initial state gives control to the main menu, which is the central point of navigation for the solution. From here, the user can access various options. The “New Game” option further triggers the “Game creation” event, which is where the user may set up initial parameters for the start of a game (number of players, player icon etc.). This event, along with the “Load Game” event, which read a saved game file from disk, then lead to the gameboard event, which is where the majority of actual gameplay will take place. Once the user is done playing, they can then trigger the save game event. The titbits and options events both open their own pages; the options event allows users to tweak their user configuration settings, and titbits allow users to review pieces of synthetic biology they may have picked up while playing the game. The latter of these was especially inspired through concerns shared by prior students during interviews in the requirements elicitation process. Both events lead back to the “Main Menu” event. Finally, the “Exit” and “Save Game” events both lead to the end of application use and thus flow control to the final state.

### 3.5 Low Fidelity Prototype

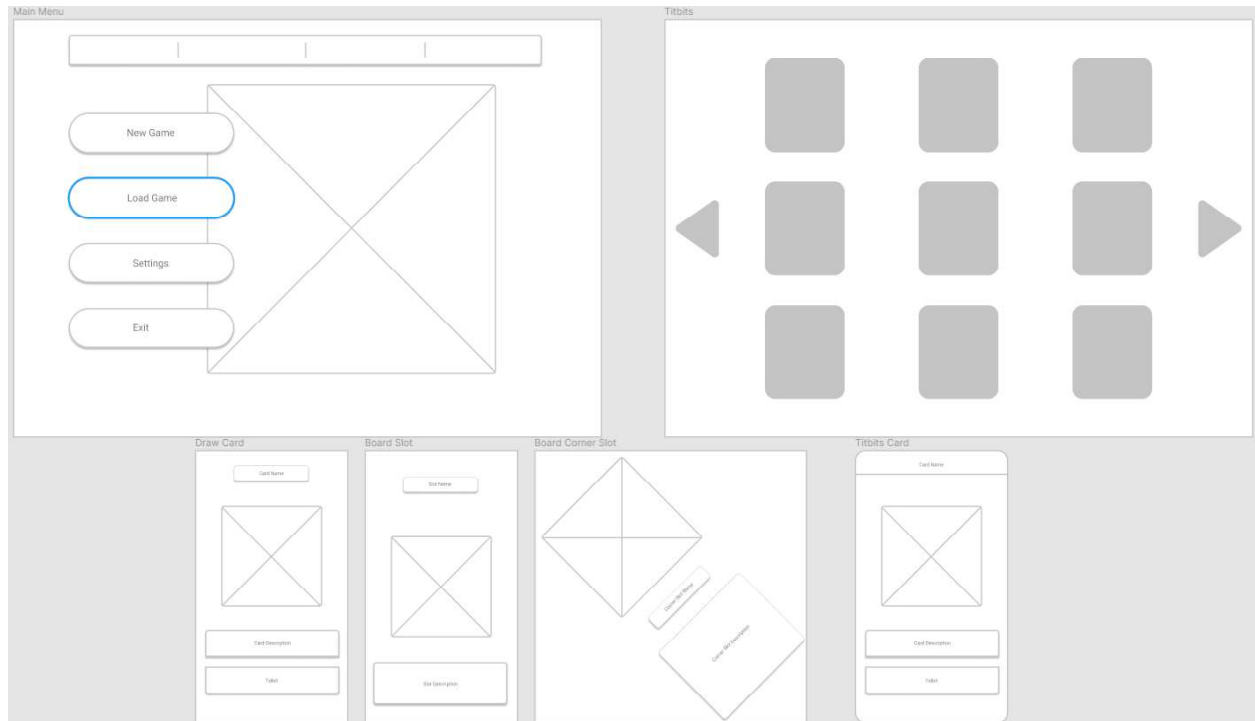


Figure 3.3: Low Fidelity Prototypes

The final prerequisite to development was the creation of low fidelity mock-ups that would provide the first visual representations of how the final solution could appear. These were malleable and prone to frequent change even during development but also allowed for high-level interface testing. As shown in the above image, many important events from the activity diagram translated directly to screens that required their own separate designs. Also, designs were made for the various blocks that make up the gameboard. This approach allowed many different design ideas to be explored quickly and at low cost, which streamlined the development process as it was much easier to visualise what needed to be achieved.



## **Chapter 4: Implementation**

### **4.1 Technologies Used**

#### **4.1.1 Figma**

This is a prototyping tool developed by Figma, Inc. that is used to create vector graphics and prototypes of various fidelities. In this project, the tool was used to generate some graphics, tables, blueprints and mock-ups in service of many architectural aspects of the development process. This was very helpful in planning out some aspects of the game (e.g. gameboard layout, menu layouts etc.) and in creating graphics that could directly feature in the final solution.

#### **4.1.2 Godot**

The Godot software is a tool initially developed by Juan Linietsky and Ariel Manzur, which serves as the main developmental environment of this solution. Godot is a game engine which means it serves a similar purpose to what an Integrated Development Environment (IDE) does in conventional software engineering. This is where the construction of architectural designs and writing of code and game logic is done. For this solution, Godot's own GDScript scripting language is used for development. The language is described as “a high-level, dynamically typed programming language used to create content” [7] with similar syntax to Python, which has many helpful tools and functions that are helpful to the development of the solution.

### **4.2 Key Components**

#### **4.2.1 Nodes**

Nodes are the building blocks of any projects undertaken in Godot and are integral to the development of this solution. Nodes are very similar to objects in more conventional software solutions as they have names and editable properties that define them and can be extended to add

more functionality. A key point about how nodes work is that each node can have one parent and/or many children. This creates a tree structure that is integral to how the solution functions.

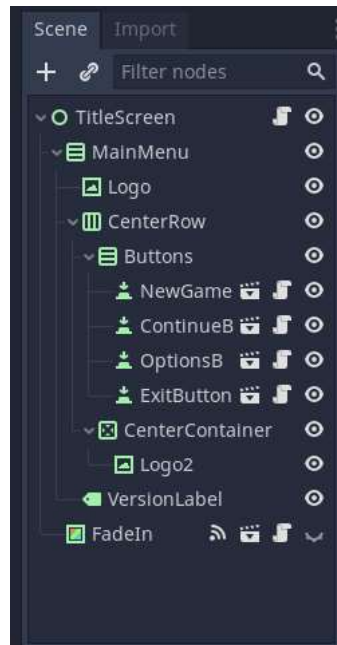


Figure 4.1: Main Menu Node Tree

The figure above shows the node tree structure for the game's main menu. The root node (node with no parents) is named TitleScreen. Much like objects, most nodes have been given more meaningful names to help guide development. The tree structure is important as, amongst other things, it determines the order in which nodes are drawn to the screen on execution which can affect application performance. Each node can have scripts attached to it to outline its functionality, as seen with the buttons, where each has scripts attached to aid in switching scenes (more on scenes to be discussed) when clicked. Signals are also a powerful tool node are capable of utilising. As the name suggests, nodes can “emit” certain signals to indicate when certain criteria are met. In this tree, we can observe that the ColorRect node, renamed “FadeIn”, emits a signal. This signal is actually a custom signal called fadeInFinished, which is emitted when a different signal is

emitted by the animation player attached to it, letting it know the animation has played. This `fadeInFinished` signal is detected in the script for the root node, telling it to switch scenes. All these together create a transition effect as one scene is switched to another, which is just one of the many ways the nodes interplay in the solution.

## 4.2.2 Scenes

Scenes are the next logical step from nodes. In actuality, a tree of nodes, as we have just seen, is, in fact, a scene. Scenes always have one root node. As shown in figure 3, they can be saved to disk and reloaded as needed, but most importantly, they can be saved instanced much like an object might be. For this section, it may help to have a more visual representation of the scene

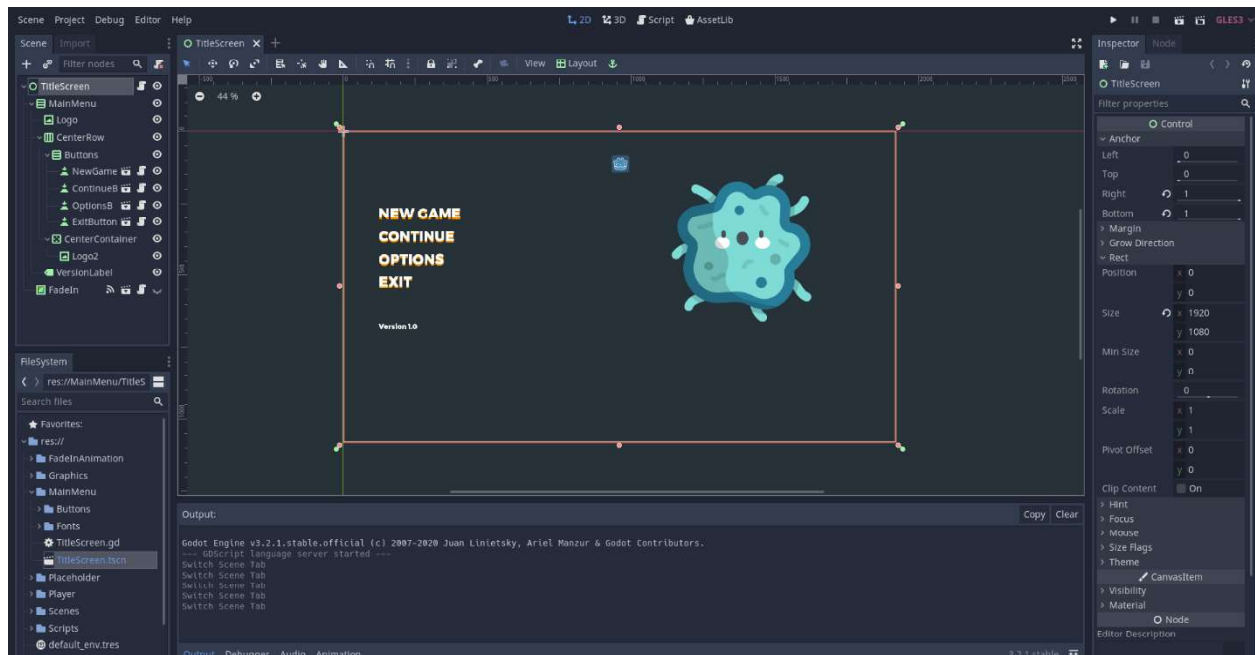


Figure 4.2: Main Menu Scene

### **4.2.3 Resources**

Resources refer to the data container tools that exist in Godot. These can include a range of anything that involves storing data to the client's disk from game assets, all the way to scenes themselves which can be considered a resource. There are external resources that are saved singularly as their own files and built-in resources which are saved inside other files, often the .tscn and .scn files that scenes are stored as. These resources are only ever loaded once when the application is in use, and once the resource is no longer in use, the engine automatically frees it. The main idea here is that nodes often rely on resources to provide some data or information to aid in their functionality. Resources play a crucial role in the save structure of the game, which is discussed later in this chapter.

## **4.3 Key Implementation Techniques**

### **4.3.1 Player Movement**

Player Movement may have seemed like one of the simplest aspects to implement in a game like this, but it quickly became one of the most complicated. This was largely due to the plethora of approaches that could be taken to achieve player movement in Godot. From linear interpolation to the `move_toward()` function to the use of the `AnimationPlayer` node, careful consideration was necessary to determine the right approach.

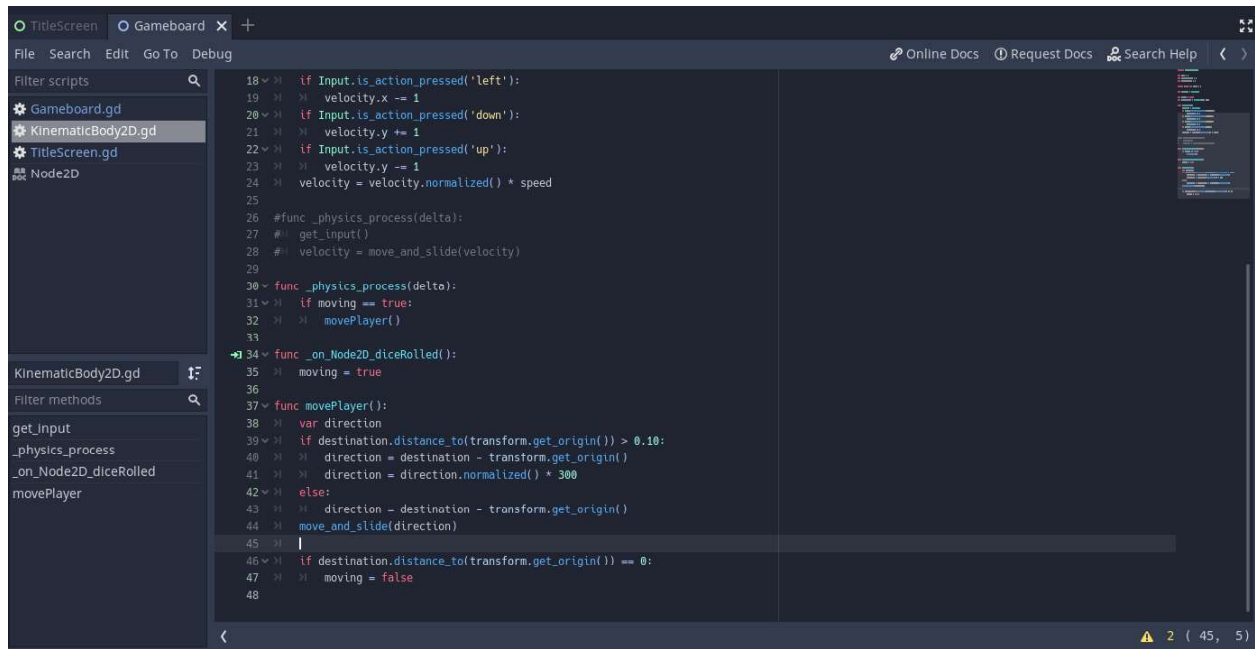


Figure 4.3: Early Player Movement Code Sample

As shown in the figure above, the approach chosen in the end was the use of the `move_and_slide()` function. This is a type of movement that is particularly used for objects of the `KinematicBody2D` node as well as its 3D counterpart. When the dice are rolled, it triggers a signal which connects to the `_on_Node2D_diceRolled()` function, which sets the moving variable to true. This is important as player movement needs to be tied to a `_process()` or `_physics_process()` function. These functions are called and executed each frame (one single image in the sequence of images constantly displayed to the screen). We use the `_physics_process()` function in this case as it executes later in the main loop cycle and executes at regular delta steps, delta being a variable passed by the engine, which is meant to represent the length of time since the previous frame displayed. Consequently, the `_physics_process()` function allows for more accurate state variables for physics objects (such as the position of the object which we get using the `get_origin()`) to be retrieved. The `movePlayer()` function is then called, which uses the player's current position and

destination position to calculate the direction in which to move. Finally, the `move_and_slide()` function is called to enact the movement to the appropriate tile. The moving variable is then set back to false, indicating that the current movement is complete and preparing it for the next movement.

### 4.3.2 Player Queue (Turn-based Gameplay)

Conceptually the player queue was relatively simple to plan out but required some implementation in practice. The player queue refers to the structure that allows different players on the same board to carry out their own action in their own separate atomic turns, one after the other. Queues are often implemented using arrays, and this was the approach chosen here. However, instead of just explicitly declaring an array, a more abstract approach was taken. It was decided that a group of player nodes would instead be created as children under one parent `Node2D` node that would act as the controller of the queue. Doing this, we could then give control to one child (player) at a time and then move on to the next when the child's turn is complete.



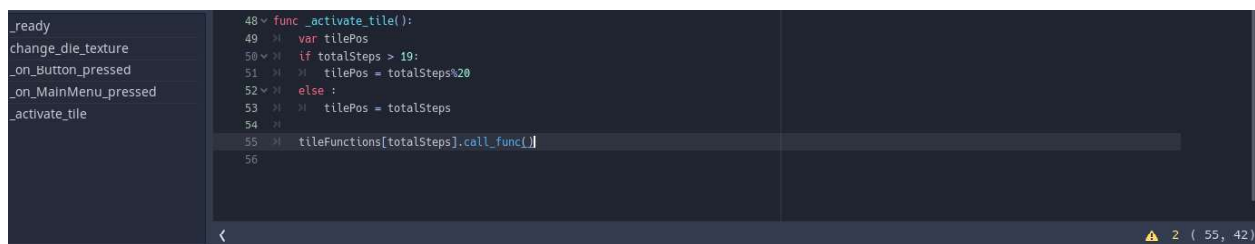
Figure 4.4: Early Player Queue Code Sample

As shown in the preceding image, the key method in this implementation is the `yield()` function, which provides us with the ability to utilise the power of coroutines to facilitate the turn-

based system. We first create an `initialise()` function for the queue, which can be called by the board (the queue's parent node) when the game first starts to set the first active player. In our `next_turn()` function, we can then use the `yield()` command to wait for the active player to complete its turn, after which we can then move down our player list and set the next player as active

### 4.3.3 Board Slot Actions

When initially planning out how actions for the various slots on the board would be carried out, the first major issue that needed to be addressed was how player position on the board would be tracked. Godot includes many node structures for dynamic interaction between game objects. The `KinematicBody2D` node and `CollisionShape2D` nodes were considered for this in the beginning. These two nodes are very commonly used in Godot development for basic movement and collision detection; however, they take a more physics-based approach. This means when objects collide, the `KinematicBody2D` node responds using either with a `move_and_collide()` method, which would cause the object to stop when a collision is detected, or a `move_and_slide()` method, which would cause the object to slide along the other. These properties, although useful for moving the player specifically, are not ideal for this context as we need the player to be able to insert itself inside these slots, so although the `KinematicBody2D` node is used earlier for player movement, `CollisionShape2D` was not used. Hence, a more abstract approach was developed.

The image is a screenshot of the Godot 4.2 code editor. On the left, a vertical list of signals is visible: `_ready`, `change_die_texture`, `_on_Button_pressed`, `_on_MainMenu_pressed`, and `_activate_tile`. The `_activate_tile` signal is selected. The main editor area displays the following GDScript code:

```
48 func _activate_tile():
49     var tilePos
50     if totalSteps > 19:
51         tilePos = totalSteps%20
52     else :
53         tilePos = totalSteps
54
55     tileFunctions[totalSteps].call_func()
56
```

The code is written in a dark-themed editor with syntax highlighting. At the bottom right of the editor, a status bar shows a warning icon, the number '2', and the coordinates '( 55, 42)'.

Figure 4.5: Early Slot Action Code Sample

Firstly, a new `totalSteps` variable was added to the player object. This variable would keep track of the total number of steps taken by a player along the board. During the first pass along the board, this variable would indicate the slot the player would currently be standing on (the player position) and on subsequent passes along the board, the calculation `totalSteps%20`, provided a way to always ascertain the player position and hence the slot that had been landed on. The number 20 was used as there are 20 total slots on the board. Functions for the various behaviours of these slots were then created and were stored in a dictionary using the `funcref()` function, which allows us to save functions as variables. This dictionary could then be indexed based on the player position value obtained, triggering the appropriate response.

#### 4.3.4 Save Structure

As mentioned earlier, resources play a key role in developing the save structure for the game. Specifically, we use the Godot File object, which is a resource that allows us to permanently store data on a user's disk and read this data back when it is needed. Files can be used for many purposes; however, for this project, the main applications identified were for the implementation of a save data module and the implementation of user configuration settings, the former of which was prioritised as user configuration falls into the non-functional requirement category and hence could be added later.

There are two key functions that come into play here; the `save()` method, which handles the writing of data, and the `load()` method, which handles the reading of data. A dictionary is used to save the key object and variables at the current state of the application and when load is called, these are read from memory and used to draw the canvas and setup the game.



## Chapter 5: Testing and Results

### 5.1 Component and System Testing

Testing for separate modules was done concurrently as an agile approach was used for development with integration of these features still being worked on. This is elaborated on in the requirements section of the Chapter 6.

### 5.2 User Testing

User testing took priority in this project as the aim was to create a solution that was catered towards the user. To properly test and assess the quality of the proposed solution, two forms of testing were developed, the results of which are below. Two test groups with varying backgrounds but with demographics similar to those surveyed were selected for testing.

Table 5.1: Usability Testing Results

Tester	Q1	Q2	Q3	Q4	Q5
Tester 1	Maybe	Yes	Yes	4	A lot of the buttons have scary robot faces, could that change
Tester 2	Maybe	Somewhat	No	1	More colours
Tester 3	Yes	Yes	Yes	1	Improved graphics
Tester 4	Yes	Somewhat	Yes	1	A short tutorial would be a nice touch

The table above shows the results from the usability testing, the goal of which to discern just how users felt using the application. The outcome was generally quite positive with a few things coming up that will be addressed in the following chapter, six. It would seem most users'

issues lay in visual problems. This is not a problem as this version of the prototype heavily neglected this in favour of functionality, hence, such responses were somewhat expected. The questions used for this can be found at appendix A.3.

Table 5.1: Effectivity Biology Quiz Results

Tester	Initial Score	Final Score
Tester 1	20%	50%
Tester 2	10%	60%
Tester 3	10%	20%
Tester 4	10%	30%
Tester 5	20%	50%
Tester 6	30%	70%

The table above shows the performance of testers on the Synthetic Biology before and after they were given time with the application. These results show a clear difference. The initial results show testers averaging a very low value of about 16%; however, final results, although not exactly perfect, show a marked improvement of 30% on average. This serves as a bit of proof of concept for this entire project as even with a limited amount of time interacting with an early prototype of the solution, testers show a noticeable improvement in a quiz that was designed to test foundational information found in the game, along with a few less familiar questions to test if their intuition or interest in the field had also been stimulated. These are quite positive findings.

## Chapter 6: Conclusions and Recommendations

### 6.1 Requirements

Functional Requirements	Current Progress
The user should be able to visualise the subject matter when playing the solution.	In Progress
The solution should have a reference point where users can review material learned.	Achieved
The solution should deliver approved synthetic biology subject material as they play.	Achieved
The user should be able to save specific pieces of information they come across and consider important to their device.	In Progress
The user should be rewarded for progressing through the solution and retaining information.	Not Achieved

The table above shows how much progress was made in addressing the functional requirements outline earlier in this report. The first requirement will be addressed more with changes to design concepts discussed more in the future developments section. As is shown, the second and third were achieved largely through the implementation of the titbits section of the game. The third requirement is only partially accomplished by this as although titbits store key pieces of information, the user does not have direct control over what is stored. There is currently no reward system which results in the fifth requirement not being achieved; however,

considerations are being made to adjust the number of titbits shown based on user playtime to address this requirement.

## **6.2 Current Limitations**

The three main limitations of the current solution implementation come in the form of game rules, feature isolation and aesthetics. The evolution of the game rules/logic will be expanded upon later. However, the primary issues boil down to it being a bit too simplistic for testers. A new structure with a necessary complexity needs to be developed to retain users' attention better and for longer. Feature isolation is also a bit of an issue as during implementation, some features, although somewhat functional, were developed in isolation of each other; hence, more work needs to be done to adapt and integrate these with each other. This need to be done in a way that creates a better, more cohesive system but does not necessarily compromise the modularity of the solution. The final limitations are simply aesthetics. As functionality was prioritised, the design for the game ended up being relatively low quality. Game assets require higher quality textures and some UI and UX elements require major overhauls. In the future, these design aspects may be outsourced to better-skilled designers to ensure a high-quality experience is developed for the user. This is key as it related directly to non-functional requirement 5.

## **6.3 Future Development**

### **6.3.1 Animation**

Animation often refers to the creation of moving images on the screen. Leveraging these can be a necessary or supplementary part of an application on a case by case. For this project, animation did not directly affect the functionality and hence was not prioritised for the developed prototype. Although we observe the player move along the board, this was not achieved using

explicit animation tools and rather, the only explicit use of animation was done during transitions between some scenes.

Future iterations of the project would seek to improve on this by providing more animation for various aspects of the game, such as rolling dice, UI navigation, drawing cards etc. This actually plays a role in addressing functional requirement one and non-functional requirement two by making it more obvious to the user when events take place and helping them to really envisage what they are doing.

### **6.3.2 Shaders**

Shaders are another tool that could be used to enhance the visual aspects of the game in pursuit of fully achieving requirement 1. They are described as programs used to render pixels and are often used for “detailing shadows, lighting, texture gradients, and more”. Introducing these to the game would allow for the creation of new visual effect that, in hand with more robust animation, can alleviate the usability of the application and the user experience as a whole.

### **6.3.3 Game Logic Evolution**

As functional as the current prototype of the game is, a key point that was both noticed during development and subsequently echoed by the test group was how the solution felt like a game of chance. At the current stage, this may not necessarily be a bad thing as the prototype is based on a physical version with its own limitations; however, moving forward, it would be ideal to alter the rules of the game to provide more interactivity to the application and users. This would aim to improve application retention and replay value (the capacity of a game for continued play after the first completion). Proposed changes are presented by, but not limited to, the points below:

- Introduce special attributes to different variants of each type of plasmid part that affect how the constructed plasmid functions. For example, Promoter part cards could have a “cloning value”, which would affect how well the plasmid could be cloned.
- Introduce an “energy” system where different plasmid parts and part variants cost varying amounts of energy causing users to carefully consider whether or not to acquire a part card they may draw.
- Introduce antibiotic gameplay, both in terms of antibiotics that can be played against other players and new antibiotic part cards that can be added to a plasmid to provide immunity.
- Overhaul the gameboard design to allow for more game-altering board slots.

Further, testing with students/potential students and meetings with lectures will also be held to determine how best to address these and any other new changes.

#### **6.3.4 Tutorials**

It was revealed by testers that it might initially be difficult to know what to do when playing the game for the first time, and thus some guidance would be beneficial. Hence, a tutorial needs to be developed. This could start out as something relatively simple such as a new scene that just lists out the rules or a new event that actively guides the user on their first playthrough showing them what to do, similar to how modern applications increase the learnability of their products.

#### **6.4 Final Conclusion**

All in all, what this project has shown is that although it may seem quite a niche, Synthetic Biology is indeed a field that has the capacity to retain much more attention than it currently does if only people could get over their initial inhibitions; games can be a very helpful tool in fostering this change.

## References

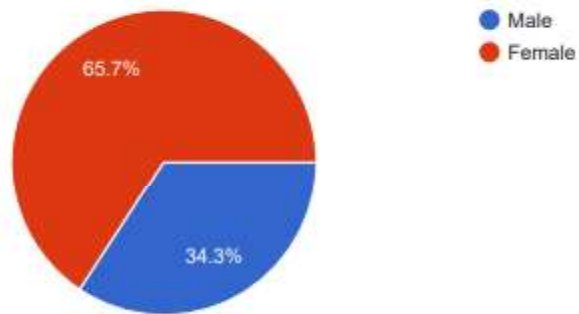
- [1]Microbiology Society. What is Microbiology? Retrieved October 14, 2020 from <https://microbiologysociety.org/why-microbiology-matters/what-is-microbiology.html>
- [2]Eleonore Pauwels. 2013. Public Understanding of Synthetic Biology. *BioScience* 63, 2 (2013), 79–89. DOI:<https://doi.org/10.1525/bio.2013.63.2.4>
- [3]Current Uses of Synthetic Biology. *BIO*. Retrieved October 14, 2020 from <https://www.bio.org/articles/current-uses-synthetic-biology>
- [4]2019. Gaming worth more than video and music combined. *BBC News*. Retrieved October 14, 2020 from <https://www.bbc.com/news/technology-46746593>
- [5]C. Aaron Price, Katherine Gean, Claire G. Christensen, Elham Beheshti, Bryn Pernot, Gloria Segovia, Halcyon Person, Steven Beasley, and Patricia Ward. 2016. Casual Games and Casual Learning About Human Biological Systems. *Journal of Science Education and Technology* 25, 1 (2016),
- [6]Understanding Basic Game Architecture | Studytonight. Retrieved November 18, 2020 from <https://www.studytonight.com/3d-game-engineering-with-unity/game-development-architecture> 111–126.
- [7]GDScript basics. *Godot Engine documentation*. Retrieved March 31, 2021 from [https://docs.godotengine.org/en/stable/getting\\_started/scripting/gdscript/gdscript\\_basics.html](https://docs.godotengine.org/en/stable/getting_started/scripting/gdscript/gdscript_basics.html)
- [8]2020. What Are Shaders in Video Games? The Ultimate Resource for Video Game Design. Retrieved April 27, 2021 from <https://www.gamedesigning.org/learn/shaders/>

# Appendix

## A.1 Survey Results

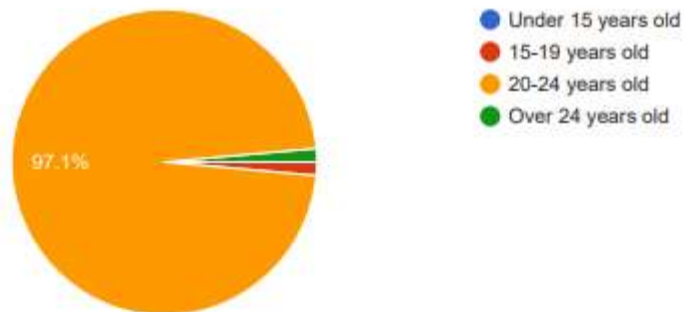
### 1. Sex

70 responses



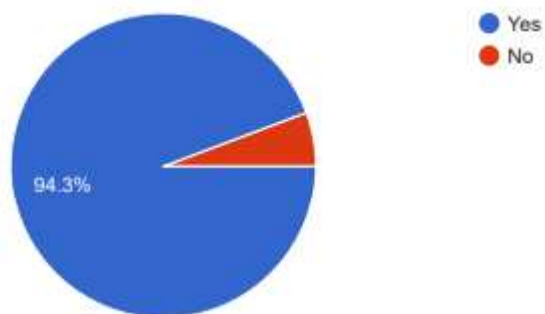
### 2. Age

70 responses



### 3. Have you ever used a game to help you learn something? (Whether intentional or accidental)

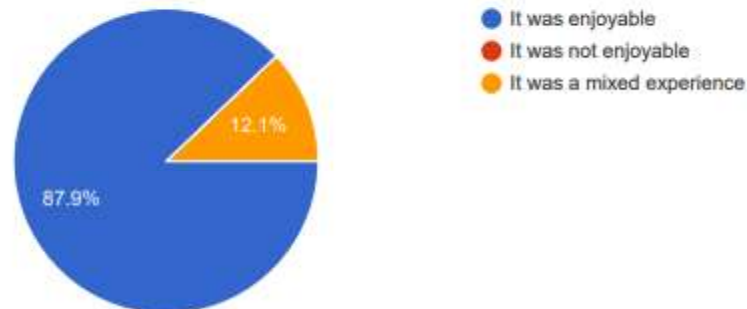
70 responses





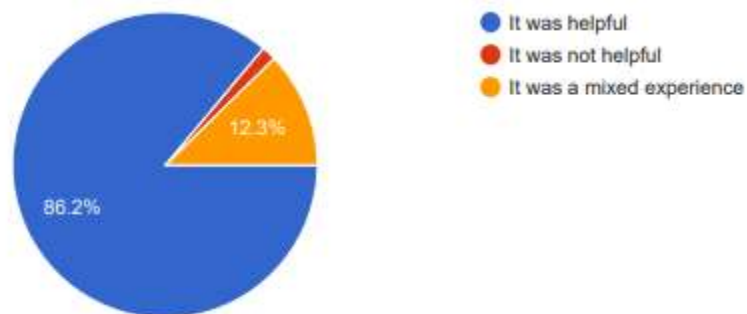
4. If you answered yes to the question 3, please state whether or not you found this experience enjoyable.

66 responses



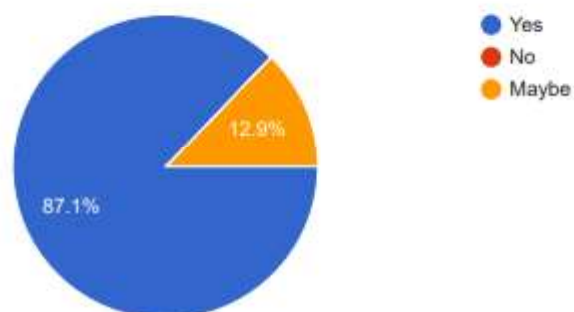
5. If you answered yes to the question 3, please state whether or not you found this experience helpful in your learning?

65 responses



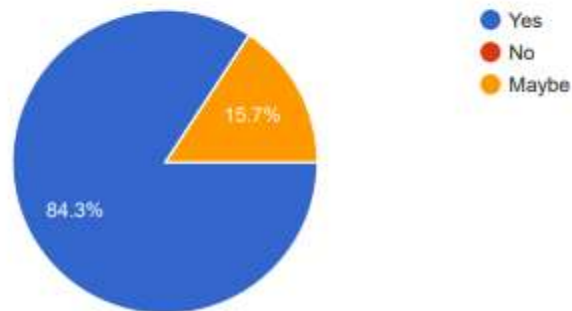
6. Do you think the use of games could be a good way to introduce yourself to an unfamiliar field of study?

70 responses



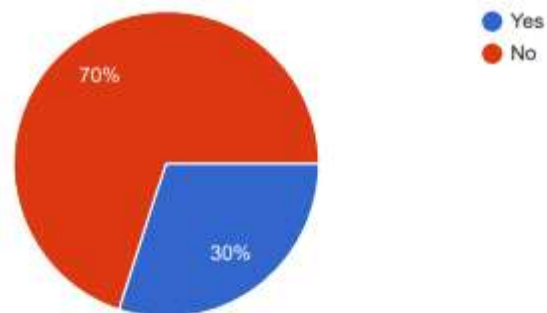
7. Do you think the use of games could be a good way to further your knowledge in a familiar field of study?

70 responses



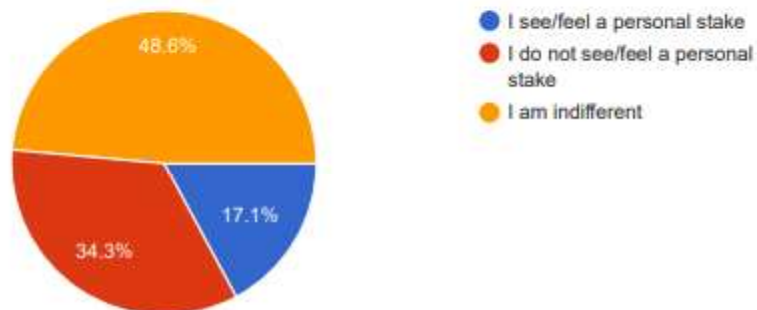
8. Coming into this survey did you know what Synthetic Biology was?

70 responses



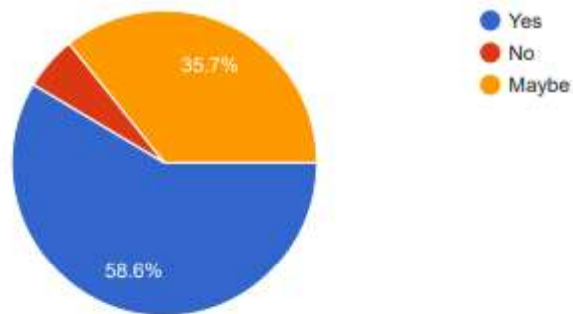
9. Do you see/feel any personal stake in the Synthetic Biology field whatsoever?

70 responses



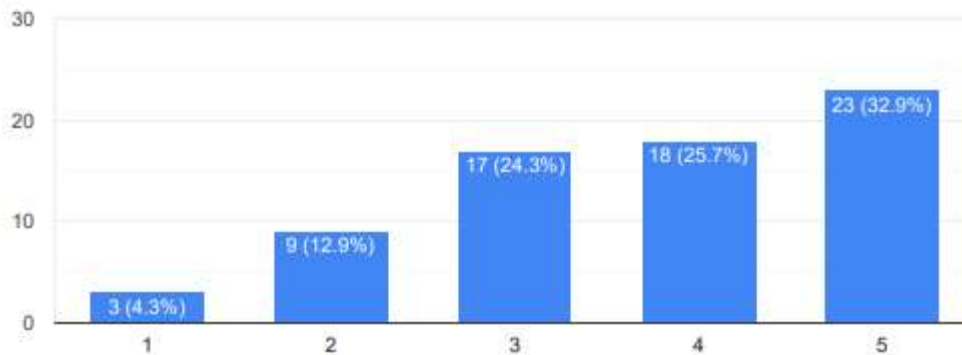
10. Do you think the Synthetic Biology field has practical value?

70 responses



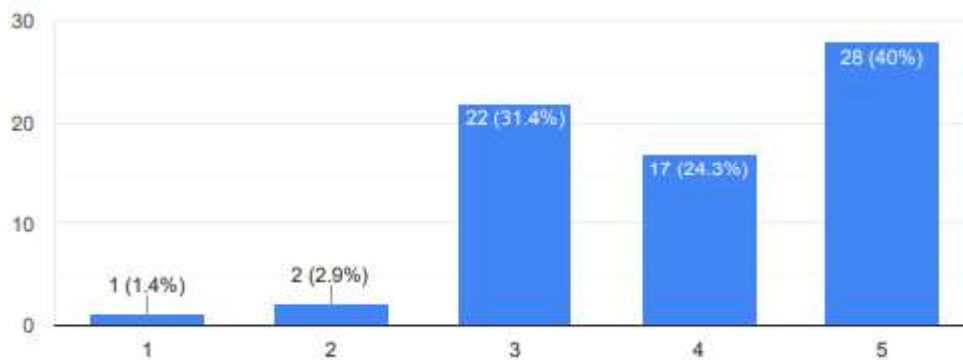
11. On a scale of 1-5 how difficult do you think Synthetic Biology is/would be for you to study?

70 responses



12. On a scale of 1-5 what is your perception of how steep the Synthetic Biology learning curve is/might be?

70 responses



## **A.2 Student Interview Guide Questions**

Name?

Gender?

Age?

Education Level?

Do you have any experience with synthetic Biology?

Why were u attracted to it?

How do u feel about your experience?

Was it challenging in any way?

How/why?

Learning curve?

Have you ever used a game to help learn something in any way?

Describe?

Did you enjoy it?

Would you be interested in such an experience for synthetic biology?

What kind would you like to see in such a solution?

What Platform would be preferable for you to play on?

### **A.3 User Testing Questions**

1. Do you feel any new interest in Synthetic Biology after your experience?
2. Did you find navigating the application easy?
3. Did the application run as informed?
4. How likely are you to check out the final version of this application?
5. What do you think could be improved about the solution?