PROJECT G - EDUCATIONAL GAME FOR LEARNING GENETIC ALGORITHM

MR. THANAPAT MAHISKHAMIN
MR. POJNARIN NANTA
MR. SIRAPOP APANANDA

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI
2022

Project G - Educational Game for Learning Genetic Algorithm

Mr. Thanapat Mahiskhamin
Mr. Pojnarin Nanta
Mr. Sirapop Apananda

A Project Submitted in Partial Fulfillment
of the Requirements for
the Degree of Bachelor of Engineering (Computer Engineering)
Faculty of Engineering
King Mongkut's University of Technology Thonburi
2022

Project Committee

..... (Assoc.Prof. Natasha Dejdumrong, D.Tech.Sci.)	Project Advisor
..... (Taweechai Nuntawisuttiwong, Ph.D.)	Project Co-Advisor
..... (Asst.Prof. Suthathip Maneewongvatana, Ph.D.)	Committee Member
..... (Jaturon Harnsomburana, Ph.D.)	Committee Member

CONTENTS

PAGE

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

CHAPTER 1 INTRODUCTION

In this chapter, we will describe the basic information of the project including background, objectives, scope of work, and project schedule.

1.1 Background

Nowadays, human technologies have been growing exponentially with no sign of slowing down; with the ongoing pandemic, many things have shifted to the digital world, disrupting older technologies and forcing people in this generation to improvise, adapt, and overcome their capabilities. One of the constantly-evolving technologies is the game industry.

In recent years, computer games have been getting more attention from the public as the game industry has been rapidly elevating nonstop. Usually, these media get acknowledged as relaxing activities; however, they also have great potential as learning tools [1]. They are easy to access by nearly everyone, especially younger generations who are accustomed to computers and technologies. Furthermore, they can easily get caught by entertainment media; games can serve as tools for education for them. There are lots of educational games coming out, and they cover almost every topic, from simple language or first-grade math to physics, chemistry, and computer science.

With the rapid growth of technologies and knowledge, education has become more radical than ever; to catch up with everything. Since the popularity of computer games is increasing, the interest in learning computer engineering is increasing. However, some content in this field is hard to study, for example, the Genetic Algorithm. Integrating knowledge about the Genetic Algorithm with computer games provides friendlier ways to access and understand the Genetic Algorithm.

An article suggests there is currently no academic game teaching about the Genetic Algorithm [2]. After some consideration, research, planning, and designing, our group proposed an educational game to educate the players, simulate how the Genetic Algorithm works, and present them to the audiences in an interactive media with the story and incentive to play. To help and motivate them to have more understanding of the said topic, we use puzzle and breeding simulator genres with the design of outcome-based education for teaching, evaluation, and education. We also use role-playing and auto-battler genres with a game design theory to motivate the players and let them have fun while playing the game. Using games as a medium for education encourages players to have the motivation and focus by giving engaging gameplay and pleasing visual and audio aesthetics to the players [3].

1.2 Objectives

As said before, these objectives are separated into two parts, those for the organizers; and those for players.

1. To study and research Unity engine, C#, and the Genetic Algorithm.
2. To make a simple educational game for people who want to learn about the Genetic Algorithm.
3. To research and apply the Genetic Algorithm in the game-making field.
4. To assist players in practicing making and planning decisions and managing resources.

1.3 Scope of Work

In this part, we will list system requirements, systems included in the project, and educational topics we will cover.

1. Windows-supported game developed using Unity.
2. In-game System
   (a) Breeding Farm
   Simulating monster breeding using the Genetic Algorithm. The system consists of breeding farms and habitats. The breeding can only be done on breeding farms.
   (b) Weapon Factory
   Using the Knapsack problem as a weapon creation. There are 5 factories in total.
   (c) Research Lab
   The system where players must learn and test their knowledge about the Genetic Algorithm to unlock other new facilities.
   (d) Miscellaneous
      i. Shop
      The place where players can spend their money to buy more monsters or weapons to use in their farms and factories.
      ii. Time progression
      The system allows players to renew the content in quest, arena and skip past the times used for the breeding process.
      iii. Money
      The currency for buying more resources for the farms and factories, fixing the broken parts and breeding monsters or weapons to make a new generation.
      iv. Arena
      The system for players to earn money by using their monsters and weapons to fight the randomly generated enemies.
      v. Quest
      The system for players to earn money by sending their monsters or weapons that match the requested monsters. The amount of money that players earn depends on the grading of how similar between the sent monsters and the requested monsters.
3. Educational topic
   (a) Genetic Algorithms
      i. Selection
         A. Random Selection
         B. Tournament-based Selection
         C. Roulette Wheel Selection
         D. Rank-based Selection
         E. Elitism
      ii. Crossover
         A. Single-point Crossover
         B. Two-point Crossover
         C. Uniform Crossover
      iii. Mutation

A. Bit Flip Mutation

B. Flip Bit Mutation

C. Boundary Mutation

(b) Real-world problems

i. Standard 0/1 Knapsack Problem

ii. Multiple 0/1 Knapsack Problem

1.4 Project Schedule

We use Scrum as a development process. We will list details about development activities, product backlog, and draft schedule.

1.4.1 Development Activities

Within the Scrum process, there are 3 important activities which are Sprint, Weekly Meeting, and Sprint Review.

1. Sprint
   All development processes are divided into nine 3-week sprints. The total estimated effort for each sprint is roughly equal to 27 man-half-days.

2. Weekly Meeting
   Meeting 15 minutes or less, repeatedly on Friday, when each member has their own responsibilities. The members report their own accomplishments, will accomplish, and Impediments.

3. Sprint Review
   Meeting at the end of each Sprint, 3 hours or less, to review completed tasks, design artifacts, and adjust Backlog and plans for the next Sprint.

1.4.2 Product Backlog

We will list the task breakdown in a form of the product backlog. This also includes the priority and the estimated effort for doing each task.

Table 1.1 Product Backlog

ID	Story	Effort Estimate (Man-Half-Days)	Priority
1	Developers need to do project documents.	36	1
2	Players can learn about the Genetic Algorithm.	36	2
3	Players can apply knowledge to solve real-world problems.	32	3
4	Players can breed their monsters.	20	4
5	Players have mandatory goals for playing a game.	12	5
6	Players have other activities to spend their time during the breeding phase.	32	6
7	Developers integrate all the systems.	20	7
8	Developers add more aesthetics to the game.	32	8
9	Developers assure quality.	20	9
Total		240	

1.4.3 Draft Schedule

The draft schedule of the Sprints will be shown as follows.

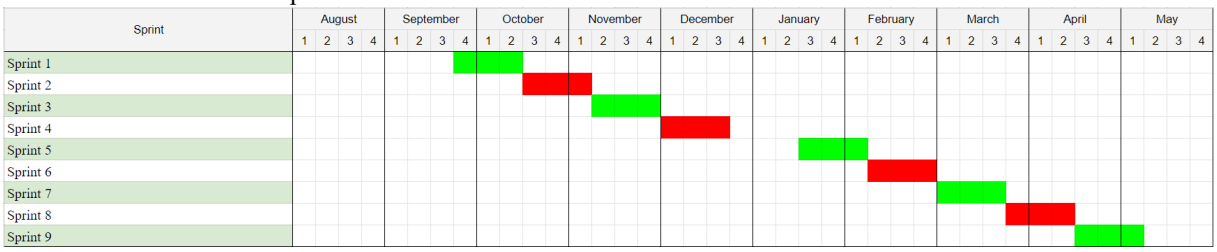

Figure 1.1: Draft Schedule of the Project

The goal of the Sprints in the first semester focus on the design. The first Sprint is for research and designing the educational content. The second and the third Sprint are for designing the game systems. And the fourth, which is the last Sprint of the first semester focuses on finalizing the design and document.

The Sprints in the second semester focus on the implementation. The fifth Sprint is designated for prototyping. The sixth and the seventh Sprint are for developing each system in the game. The eighth Sprint is planned for integrating the game. And the ninth Sprint which is supposed to be the last is for testing the full game and finalizing the document of the project.

CHAPTER 2 BACKGROUND THEORY AND RELATED WORK

For comprehension, we will present more information needed to understand our work. This chapter begins with theories involving game design, education design, and engineering content like the Genetic Algorithm. Then, we will describe the tools and technologies required to develop the game. Finally, we will discuss the works related to our project.

2.1 Game Design Theory

This topic covers the game theories we used in this project including Natural Funativity, Maslow's Hierarchy of Needs, and Eight Kinds of "Fun". We will apply these theories to increase the quality of our game in the aspect of "Fun".

2.1.1 Natural Funativity

According to Noah Falstein's theory [4], all fun derives from practicing survival and social skills, which consist of 3 categories.

1. Physical Fun
   The fun of using the physical body which is strongly related to the survival instinct. Especially under depression or threatened situations which will instantly capture the attention. A game with intense and fast action like a first-person shooter (FPS) is a good example of a game introducing this type of fun. For playing such type of game, the player needs to observe the environment using their eyes, continuously control their in-game character using their hands, and act fast to fight the enemy to win the game.

2. Social Fun
   The fun of interacting with others. The interaction can be conversations, competitions, cooperation, exchanges, etc. A multiplayer game like a massively multiplayer online role-playing game (MMORPG) is a good example of a game with social fun. In this type of game, the players continuously interact with other players in many ways through conversation, trading, or even competition. These activities sure give players a sense of social fun.

3. Mental Fun
   The fun of using intelligence. It could be achieved in many ways, such as reasoning, planning, decision-making, recognizing, solving problems or puzzles, etc. The puzzle game, a game in which the players must continuously use their reasoning, planning, or problems solving skills to progress the game, is a perfect example of this type of fun.

All the types of fun do not need to be all in a game to make it a fun game. But mixing more of these aspects of fun will help the game tend to be more fun and popular.

2.1.2 Maslow's Hierarchy of Needs

Maslow's Hierarchy of Needs implies that one's motivation has an order to satisfy based on different needs [5], representing each level in the Figure 2.1 below.

1. Physiological Needs
   It is the most essential need for every human to survive, and it is mandatory to fulfill this kind of need

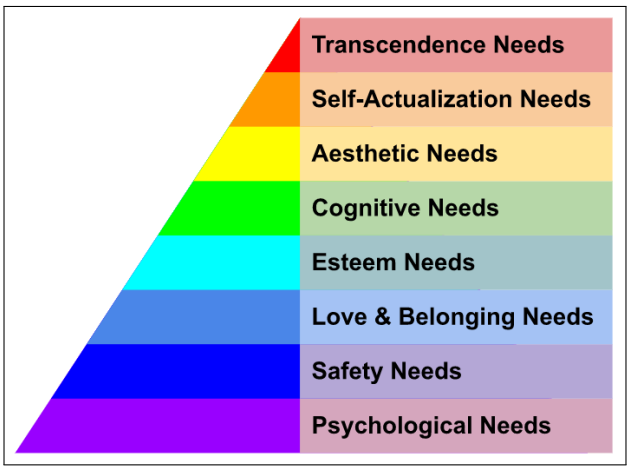

Figure 2.1: Maslow's Hierarchy of Needs

before going further, such as air, water, food, clothing, etc. This level of need may be introduced as a health point system in a game that the player must maintain to survive.

2. Safety Needs
   The need for stability in life, also regarded as the need for protection from threats, including personal security, shelter, order, law, etc. This level of need can be used as resources or items that the player could gather to make their character stronger and more stable.

3. Love and Belonging Needs
   After the previous two needs have been fulfilled, the next level of need involves a social and a feeling of belongingness. Humans need to have a sense of unity with the environment. An example includes friendship, intimacy, trust, etc. The game applying this needs as a social system like co-op, guild, or multiplayer mode.

4. Esteem Needs
   The esteem needs consist of two aspects, esteem for oneself and the desire for reputation or respect from others. This kind of need can be fulfilled by receiving dignity, achievement, status, prestige, etc. The achievement system of the game is a system directly derived from this level of needs.

5. Cognitive Needs
   The need for learning, and gaining intelligence. This includes skill, knowledge, experience, understanding, curiosity, exploration, etc. The in-game character with a skill tree or an experience system is the application of this need.

6. Aesthetic Needs
   This level of need is about seeking the thing that satisfies humans, as they need to search for and appreciate beauty, including balance, form, etc. The beautiful graphic of the game is also the aesthetic that fulfills this need of humans.

7. Self-Actualization Needs
   This need is about realizing personal potential. Causing humans to need for personal growth, and trying to be everything one person could be. Creating an immersive game that drives the player to be in a way they have not experienced before is an example of fulfilling this need through the game.

8. Transcendence Needs
   The ultimate need a person could wish. This level of need is motivated by values outside of self. This may be in a form of helping others or an ability to perform something mystical that a typical person can't do. The fantasy gameplay in which players can use magic; the famous player who helps other players in the game. These are examples of this need in the aspect of the game.

2.1.3 Eight Kinds of "Fun"

Marc LeBlanc proposed the idea of different kinds of "Fun" as a set of vocabularies to describe the kinds of fun that people are experiencing to be more specific and precise [6].

1. Sensation - Game as sense-pleasure
   This kind of fun involves the physical senses of players, including visual, sound, sometimes physical movement, or even game pace.

2. Fantasy - Game as make-believe
   It also acknowledged as escapism or immersion. It was gained by immersing oneself into the game world and possessing the ability to do things that cannot be done in the real world.

3. Narrative - Game as drama
   It revolves around unfolding stories that the game has to offer. With narration, players get a sense of direction they can look forward to, sometimes even with their narration.

4. Challenge - Game as obstacle course
   Overcoming courses of obstacles, solving difficult puzzles, defeating difficult enemies, or anything that provides highly competitive value to the player so they can challenge themselves.

5. Fellowship - Game as social framework
   This kind of fun comes from gaining social interactions and cooperating with other players. A multi-player game is a good example.

6. Discovery - Game as uncharted territory
   Exploring new things in the game world or within oneself could be fun. Players could explore new area of maps, search for secret items, or dive through side stories.

7. Expression - Game as self-discovery
   This kind of enjoyment comes from getting a chance to express oneself creatively, such as role-playing as a character in RPG games or conveying one's thoughts into creation in sandbox games.

8. Submission - Game as pastime
   Doing daily tasks repetitively and playing the game casually, which are the opposite side of Challenge, also count as fun. Sometimes, just simply enjoying a game and relaxing are enough.

2.2 Education Design Theory

This topic covers the education theories we used to design the education part of the game. We will apply these theories to ensure that our game will surely educate the player. Theories involved in this topic are Bloom's Taxonomy, Outcome-based Education, and the ADDIE model.

2.2.1 Bloom's Taxonomy

The theory published by Benjamin Bloom presents the six levels of learning [7]. These levels indicate the depth of learning and are used widely in the field of education. Each level of learning consists of a set of action verbs, the verb indicating whether the learner reaches the learning level. We use this theory to design the learning outcomes of the game.

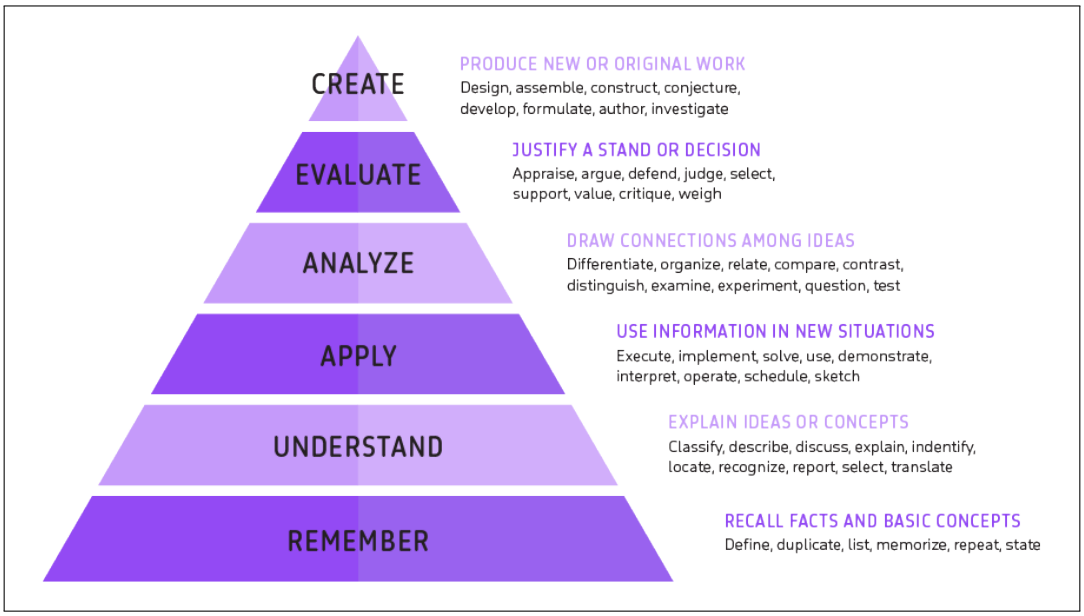

Figure 2.2: Bloom's Taxonomy with Action Verbs [7]

Figure 2.2 shows all the learning levels and their action verbs. Each of them will be described below.

1. Remember

   The lowest level of learning. The learner who achieves this level will be able to remember and recall the basic concepts of the knowledge. The action verbs in this level include duplicate, list, memorize, repeat, state, etc.

2. Understand

   The learner is able to explain the concepts they have learned. The action verbs in this level include classify, demonstrate, explain, identify, illustrate, etc.

3. Apply

   The learner is able to apply the knowledge to new situations they never met before. The action verbs in this level include implement, solve, use, operate, model, etc.

4. Analyze

   The learner is able to compare the different knowledge and see connections among the ideas. The action verbs in this level include compare, contrast, distinguish, etc.

5. Evaluate

   The learner is able to use the knowledge to judge the value of the idea and justify the decision. The action verbs in this level include appraise, judge, value, etc.

6. Create

   The highest level of learning. The learner is able to assemble and produce a new work using the combination of previous knowledge. The action verbs in this level include build, develop, formulate, etc.

2.2.2 Outcome-based Education

Outcome-based education (OBE) is an education concept focusing on the learner's ability to perform the desired task at the end of the learning or outcome [8]. The curriculum is designed backward, starting from the outcome instead of what will be taught.

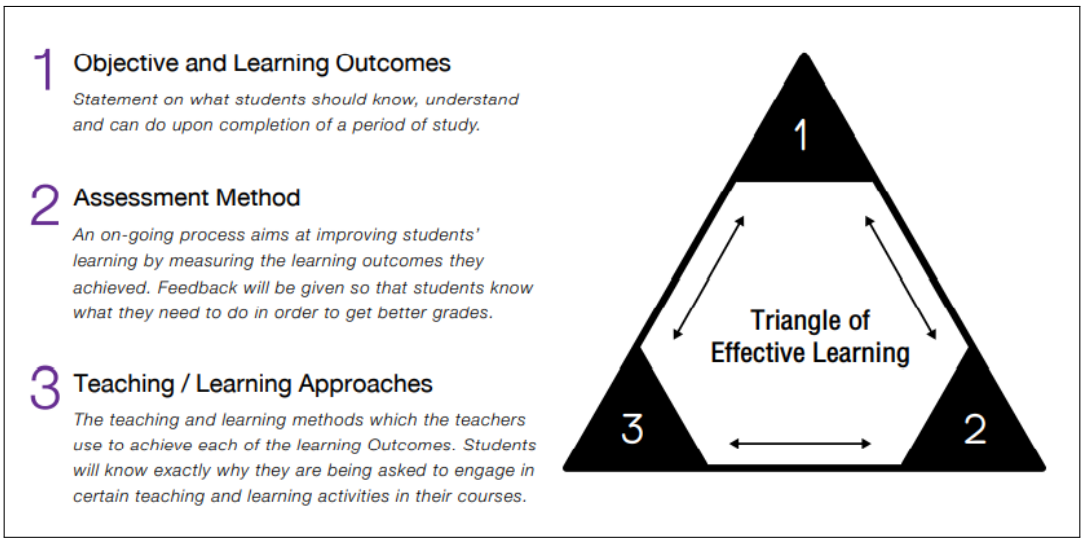

Figure 2.3: Triangle of Effective Learning [8]

According to the triangle of Effective Learning in Figure 2.3, the first step of designing is setting the learning outcome, which are skills we wish the learner to gain. The general format of the learning outcome is the action verb + object + qualifying phrase. An action verb can be derived from Bloom's taxonomy. A good learning outcome should match the SMART(TT) characteristics [9], which is an abbreviation for

- Speak to the learner: The outcome should specify what the learner will be able to do.
- Measurable: The outcome indicates how it will be assessed.
- Applicable: The outcome addresses the way the learner uses the gained skill.
- Realistic: The learner should be able to demonstrate the skill addressed in the outcome.
- Time-bound: The outcome should set the specific duration of the learning.
- Transparent: The learner can easily understand the outcome.
- Transferable: The skill in the outcome could be used in a wide range of contexts.

After the learning outcome is specified, design the assessment method to measure the learner's ability. The assessment criteria or a rubric must be specified. There is no need for one outcome to consist of only one benchmark. A good rubric should be observable and measurable.
Finally, the teaching and learning activities are designed based on learning outcomes and assessments. This includes not only the materials but also the activities. We adopt this concept in our educational game design in the education parts.

2.2.3 ADDIE Model

ADDIE model is a model for designing and developing processes used widely in many fields, including education [10]. The model provides a systematic approach to the work process consisting of analysis, design, development, implementation, and evaluation, respectively. We apply the process of this model as the process for doing the project.

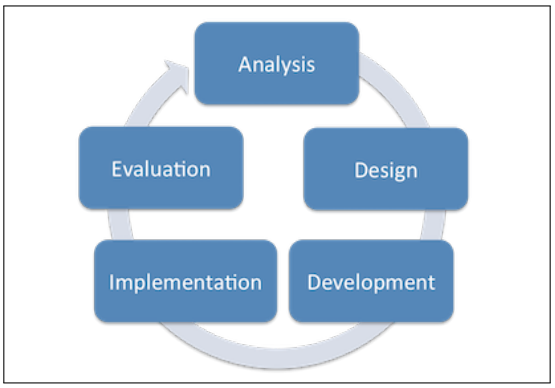

Figure 2.4: The phrases of the ADDIE Model [10]

1. Analysis
   The first step in the process focuses on identifying the problem, the learning environment, and the deliveries.

2. Design
   The main task of this phase is defining the learning objectives, evaluation tools, and content development to suit the analyzed problem.

3. Development
   After the design, the next step is developing the learning resources, including the learning material and learning activities. Those contents can be used later for the pilot test, a small-scale educational simulation used for gathering data about the feasibility of the project [11]. Such a test is applied to the group of target students to collect the necessary feedback for content revising.

4. Implementation
   This step focuses on preparing the people involved in education and implementing the actual learning resources, such as content, material, and tool that have already been analyzed and designed. In the aspect of educational games, the main task of this step can be implementing the actual computer game software.

5. Evaluation
   The last step of the model is an evaluation. The two methods of evaluation are formative evaluation and summative evaluation. Formative evaluation is the type of evaluation that is conducted at every stage of the ADDIE model and focuses on ongoing project revision. Summative evaluation occurs after the implementation, aiming to assert the learner's outcome and the effectiveness of the educational program once the course is completed.

2.3 Genetic Algorithm

The Genetic Algorithm is the computer algorithm adapted from the Darwinian theory of Natural Selection [12]. The algorithm introduces an easy-to-understand way to solve complicated problems since it's a technique with very little mathematics. Even deadlocked problems, the complex problems with conflicting conditions that require simultaneous solutions previously considered difficult can be solved with Genetic Algorithm [13].

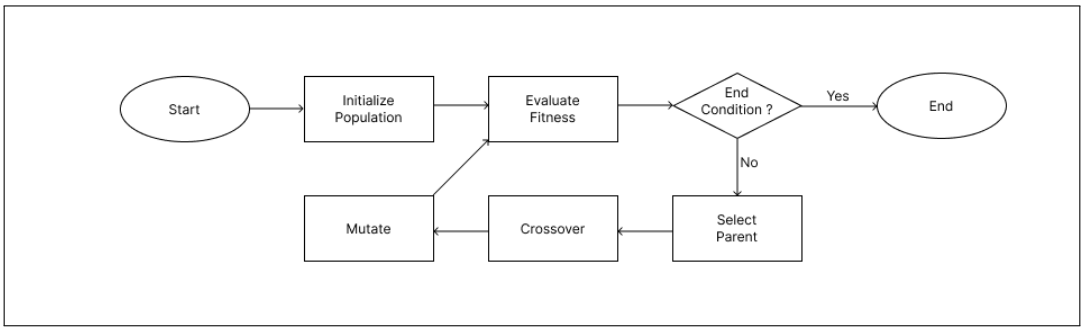

Figure 2.5: Flow Chart of Genetic Algorithm

The algorithm begins with encoding the possible solutions to the problem into a data structure called a chromosome. Such data structure usually is an array of bit values, the value that can be only 0 or 1. Other forms of data structure can be similarly used, for example, the matrix of the integer. Once the chromosome population is generated, they will be repeatedly evolved through the biological-inspired processes as shown in Figure 2.5. Commonly, the Genetic Algorithm is used to generate the optimal solutions, the best solution among a vast number of generated possible solutions, for optimization and search problems. The detail of each process in the algorithm is described as follows.

2.3.1 Population Initialization

Usually, the population initialization will be done via randomizing the set of valid possible solution chromosomes, the chromosomes that satisfy all the constraints. The population size must be large enough to maintain the diversity of the solution; otherwise, it will not be able to reach the global optimum solution in the end.

2.3.2 Fitness Evaluation

The fitness of a chromosome will be evaluated using specific methods, including fitness function and constraint. In a more general context, the fitness function is an objective function of an optimization problem which is a function that calculates the value we wish to maximize or minimize. In the context of the Genetic Algorithm, the fitness function is used to calculate the fitness value that indicates how close to the optimum solution of the chromosome is. If any chromosome violates the constraint, it is considered an invalid solution, and the fitness value of that chromosome will be zero.

In the Natural Selection rule, the survivor is one with the fittest properties. The chromosome with a higher fitness value tends to "survive", selected by the algorithm during the parent selection process. The selected parent will inherit part of its chromosome to the offspring in the next generation through the crossover process. Meanwhile, the chromosome with lower fitness is likely to be destroyed.

2.3.3 Genetic Operations

The population will be evolved by generating a new population of the existing ones through genetic operations consisting of selection, crossover, and mutation.

- Selection

  The subset of chromosomes will be selected from the current population as a parent for the next generation based on the fitness value. There are several basic ways to choose the parent, including random selection, tournament-based selection, roulette wheel selection, and ranked-based selection. Apart from the parent selection, there is also a technique that improves the algorithm called Elitism.

  1. Random Selection

     This is the most straightforward method, which is a uniformly random selection of a chromosome out of the population. The probability of being selected is not related to the fitness value of an individual chromosome.

  2. Tournament-based Selection

     This method begins by randomly dividing the population into equal groups with the number of K chromosomes in each group. Then, the chromosome with the best fitness value will be selected from each group. For example in Figure 2.6, the chromosomes are first picked as a group of 3, and the chromosome "A" is selected from the group due to its highest fitness value.

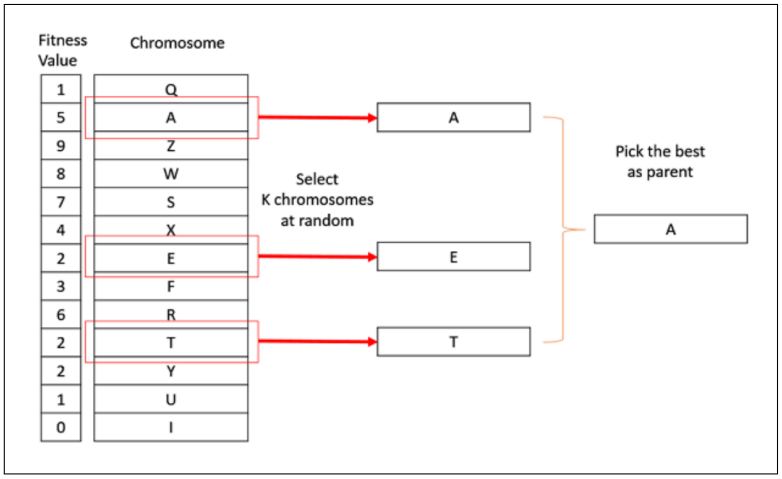

Figure 2.6: Tournament-based Selection [12]

  3. Roulette Wheel Selection

     The method is based on proportionate randomization. It can be interpreted as a Roulette Wheel where each slot represents the chance of an individual chromosome to be chosen. As the chance of a chromosome being selected proportionate to its fitness. The higher fitness of the chromosome, the larger space on the wheel.

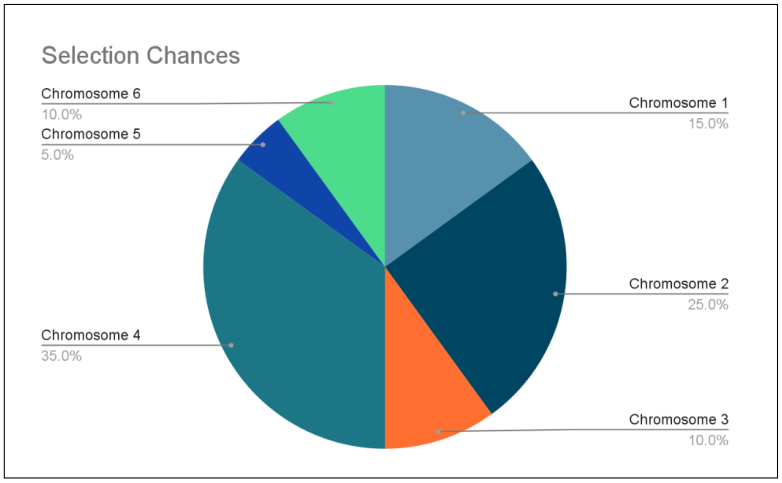

Figure 2.7: Roulette Wheel Selection

The selection chance for each chromosome is calculated using its fitness value divided by the total fitness value of all chromosomes in the population. In Figure 2.7, given that chromosome 4 has a fitness value of 7 and the total fitness value among the population is 20. So, the selection chance of chromosome 4 will be 7 divided by 20 which is equal to 0.35, or 35 percent.

4. Rank-based Selection

   The selection method is based on the rank of the chromosome. The individual in the population is ranked according to their fitness value first. Then, assign a probability to be selected proportional to the individual rank. This can be done by reassigning the fitness value of the chromosome using the reversed rank. The worst chromosome will have fitness 1, the second worst will have fitness 2, and so on [14]. The best chromosome will have the highest fitness equal to the number of chromosomes in the population.

Table 2.1: The Comparison Between Selection Methods.

Selection Method	Advantage	Disadvantage
Random	It is easy to understand and implement.	There is no use in the fitness value; the good solution might be accidentally destroyed.
Tournament-based	It can be implemented efficiently since no extra calculation like a sorting is required.	If the tournament group is large, the weak chromosome will have a smaller chance to be selected.
Roulette Wheel	The chromosome with higher fitness has more chance to be selected.	It's not fair because the worst chromosome will barely be selected.
Rank-based	All chromosomes have a similar chance to be selected which means it preserves diversity.	Computationally expensive due to the sorting process. The best solution is found slower since the difference between each chromosome is small.

All the parent selection methods have their advantages and disadvantages. The comparison between these methods [14] is discussed in Table 2.1.

5. Elitism

   Elitism is a method that helps preserve the elites, some set of chromosomes with the highest fitness value, across the generations without any change. It improves the performance of the algorithm by ensuring the preservation of the best solution. However, increasing the ratio of the elites too much hurt the algorithm due to decreasing in population diversity. From the experiment, the algorithm with a higher reliability requirement needs a higher population size but lower elitism rate [15]. So, the elitism rate is preferred to be a small percentage value around 5% to 10% [16].

- Crossover

  The crossover is the process where a pair of selected parent chromosomes exchanges part of their information to produce a new pair of offspring. There are several basic ways to perform the crossover, including one-point crossover, two-point and k-point crossover, uniform crossover, and partially-mapped crossover (PMX).

1. Single-point Crossover

   The process of this type of crossover is illustrated in Figure 2.8. The process begins with randomly selecting a single point on both parents' chromosomes as a crossover point. Then, the gene information from the right side of that point is exchanged with the other parent. The result is the pair of offspring chromosomes that inherit the gene information from both parents. This is one of the most simple crossover methods and it is also one of the lowest computational cost methods.

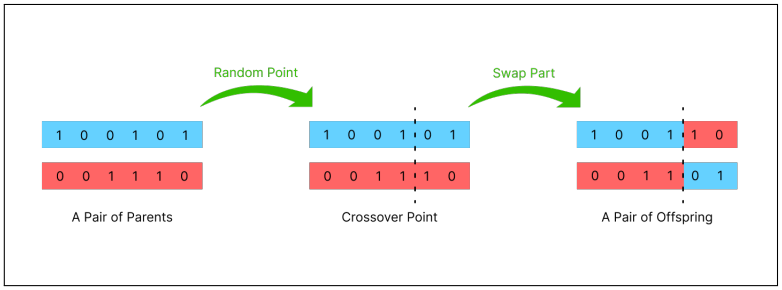

Figure 2.8: Single-point Crossover

2. Two-point and K-point Crossover

   The process of the two-point crossover and the k-point crossover is the same. The only difference between them is the number of crossover points. Figure 2.9 shows the processes of the two-point crossover. Two points from both parents' chromosomes are randomly selected to be crossover points, dividing the chromosome into 3 sections. Then, the gene information in the second section (from left to right) which lay between these two points and is swapped between the parents to create two new offspring. Note that the first and the third sections of the chromosomes which are odd-numbered sections stay still.

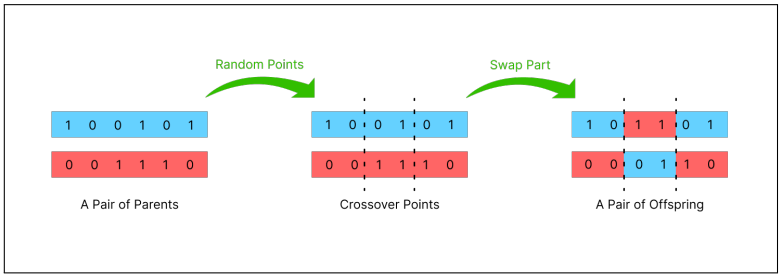

Figure 2.9: Two-point Crossover

For the k-point crossover, the k crossover points are randomized, divide the chromosomes into $k + 1$ sections. Same as the two-point crossover, the odd-numbered sections stay still. In the other hand, the even-numbered sections swapping between parents. This type of crossover help the chromosome be able to exchange its subset of information for multiple times.

3. Uniform Crossover

   In this type, as shown in Figure 2.10, the offspring are created by randomly swapping every piece of information in the parent' chromosome. The chance of switching each point is independent of another. This type of crossover allows the offspring to inherit from both parents equally and independently.

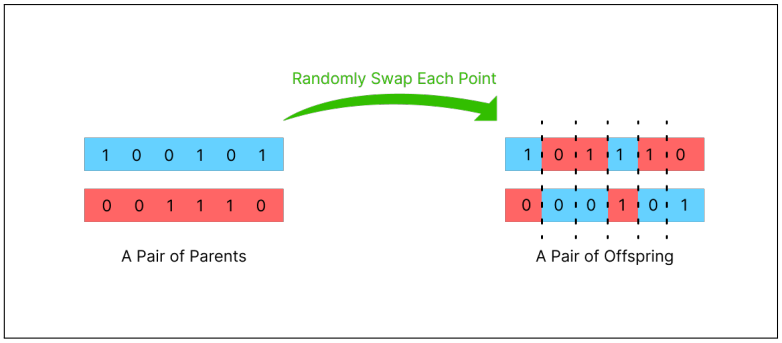

Figure 2.10: Uniform Crossover

4. Partially-Mapped Crossover (PMX)

   Whereas other previous methods cannot be implemented on the chromosome that can contain a duplicate value, the PMX can. To be accurate, PMX is a crossover method that only is used with the chromosome with such a condition. The method begins with randomly selecting two crossover points and swapping the chromosome information in the middle section. The swapped section is also called the mapping section since each information point on the parent is mapped with the corresponding point on another parent. Figure 2.11 shows an example of such mapping.

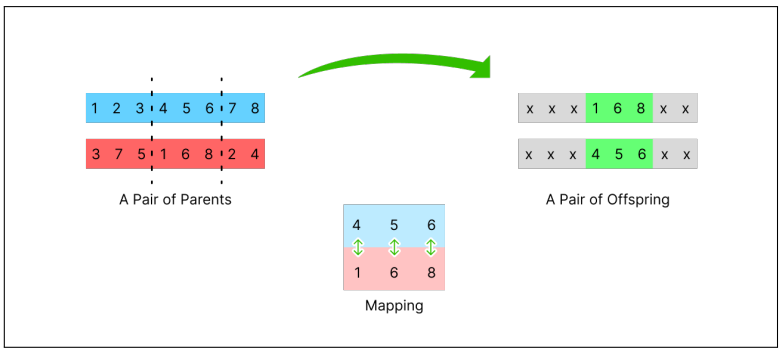

Figure 2.11: Mapping section of PMX

Then the information from its parent in the same position transfer to the offspring. If such information already exists in the offspring, that information will be changed into the corresponding value in the mapping until it turns out to be information that never exists in the offspring. In Figure 2.12, the value 1 already exists in the offspring. So, it is changed into its mapping value which is 4.

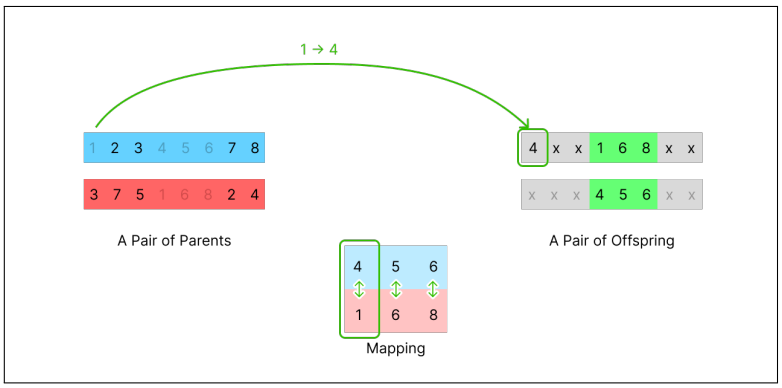

Figure 2.12: Mapping results in PMX

The rest of the information in both offspring is transferred in the same way. Figure 2.13 shows the final result of the PMX operation.

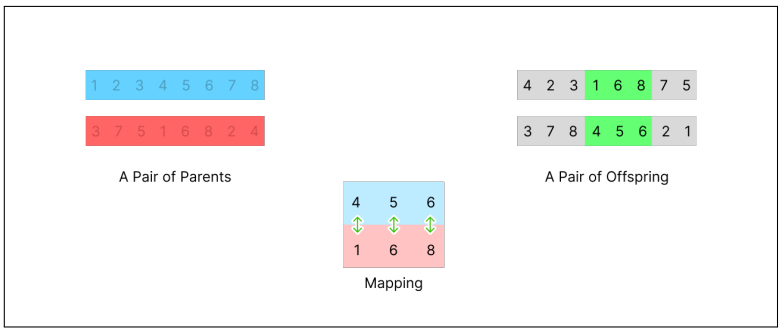

Figure 2.13: The final result of PMX

- Mutation

  A mutation is a technique of altering the gene information of an individual chromosome. The procedure of altering the information varies and depends on how the solution is encoded. By performing this process, the chromosomes in a population can get better diversity. But just like Elitism, setting the mutation rate too high makes no good to the algorithm. If we use a high mutation rate, the algorithm would act as a random algorithm that does not utilize the advantage of using other genetic operations. So, the mutation rate is preferred to be a low value similar to the elitism rate. There are different types of basic mutation including bit flip mutation, flip bit mutation, and boundary mutation.

  1. Bit Flip Mutation

     This type of mutation is used to alter the binary bit string chromosome, the chromosome consisting of only 0 and 1. The mutation can be done by randomly flipping the individual bit in the chromosome. The flipping of a bit can be done by changing its value from 0 to 1 or 1 to 0 depending on what the value it is before. For example in Figure 2.14, the highlighted bit is changed from 0 to 1.

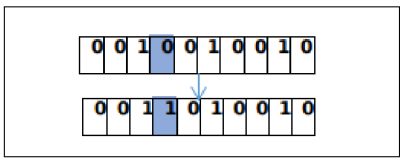

Figure 2.14: Bit Flip Mutation [12]

  2. Flip Bit Mutation

     This type of mutation is also used with the binary bit string chromosome. Performing this type of

mutation, instead of flipping an individual bit in an individual chromosome, the whole chromosome is randomly selected and every bit within the chromosome will be flipped.

3. Boundary Mutation
   This type of mutation suits the chromosome with an integer or float value. The process can be performed by replacing a specific value in the chromosome with the new random value of a lower or an upper range.

Table 2.2: The Comparison Between Mutation Methods

Mutation Method	Amount of Changed value	Suitable Type of Chromosome
Bit Flip	Randomized individual value within a chromosome	Binary bit string
Flip Bit	All information in randomized chromosome	Binary bit string
Boundary	Randomized individual value within a chromosome	Integer or float

Each mutation method suits a different type of chromosome and changes a different amount of value. Table 2.2 shows a brief comparison of each method.

2.3.4 Knapsack Problem

The real-world problem involving our project is the knapsack problem and its variant, it's one of the most popular optimization problems that can be solved using Genetic Algorithm [17]. To be accurate, the problems include Standard 0/1 Multidimensional Knapsack Problem (MKP) and Multiple 0/1 Multidimensional Knapsack Problem. The word 0/1 indicates that the problem can be encoded as a binary bit string chromosome as the solution shown in [18]. The setting and goal of each problem will be described below.

1. Standard 0/1 MKP
   Generally, the standard knapsack problem is a problem that deals with packing items in a bag while maximizing the value of the bag. Each item has its weight and value, and the bag can hold some amount of maximum weight. This maximum weight can be interpreted as a constraint that the solution must follow. The solution is invalid when the total weight of the picked items exceeds the maximum weight of the bag. The word Multidimension indicates the dimension of the knapsack or the number of constraints. For example, the Standard 0/1 MKP that has only the constraint of the weight limit can be described as only 1 dimension knapsack. The binary bit string chromosome of length equal to the number of items can be used as a solution representation. Using this type of chromosome, the value of each bit indicates whether the item with the corresponding index is selected or not.

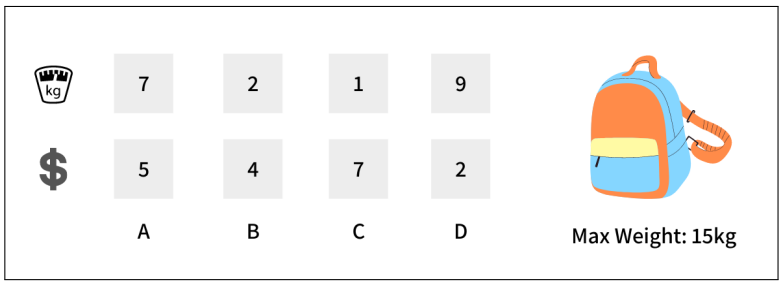

Figure 2.15: Example of Standard Knapsack Problem Setting [18]

Figure 2.15 shows the example of Standard MKP. There are four items which are A, B, C, and D. Each item has a weight of 7, 2, 1, and 9; and has a value of 5, 4, 7, and 2 respectively. The constraint is that the knapsack can hold a maximum total weight equal to 15.

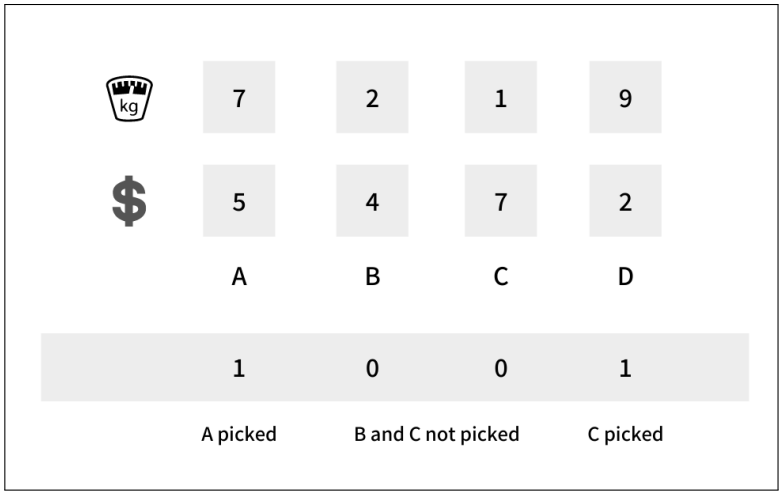

Figure 2.16: Example of Chromosome Representation of the Standard Knapsack Solution [18]

Since there are four items, the binary bit string chromosome with a length of four can be used as an encoded representation of the solution. Figure 2.16 demonstrates the chromosome with the value of 1, 0, 0, and 1. By decoding these values, we get the solution which is picking items A and D into a knapsack. Using encoding and decoding this way, we can represent any solution using the binary bit string; and we can finally use a Genetic Algorithm to solve such a problem.

2. Multiple 0/1 MKP

   The setting and the goal of the Multiple 0/1 MKP are similar to the Standard 0/1 MKP. The difference between them is the number of knapsacks in Multiple 0/1 MKP is more than one. The goal of this problem is not only the decision about whether a specific item is selected but also involves choosing the knapsack such an item should be in. The binary bit string chromosome of the length of the number of items (N) multiplied by the number of knapsacks (M) can be used as a solution representation. The first N bit in the chromosome indicates whether the item with the corresponding index is selected to be in the first knapsack or not. The next N bit of the chromosome is the indicator for the second knapsack and so on. The solution is also invalid if the same item is selected to be in more than a single knapsack.

2.4 Usability Testing

Usability testing aims to understand how users interact with the product and find issues in existing products [19]. The information gathered from the test can be used to improve the design or current development. In general, the usability testing questions are divided into 3 sets which are screening, in-test and post-test.

The screening question indicates whether the participants represent the target group. The data from this type of question includes both demographic and personal experiences. The in-test question aims to find insight into user behaviors such as the reason behind the user's certain actions. The insight from this section usually be generated during the user's interaction with the product. The post-test question aims to gather the user's additional feedback. The opinion gathered from this section can be useful for product improvement.

With the reason for the benefit-cost ratio, the recommended test user size is 5 [20]. With this size of the test user, almost all usability problems will be found while the effort of performing the test is not too high.

2.5 Languages and Technologies

We will use multiple programs and platforms to complete this project, including Unity, the game engine of choice, visual studio, the code editor compatible with Unity, C#, the programming language for Unity, GitHub, the version control platform, and Figma, for model creation.

1. C#

   C# (pronounced "See Sharp") is one of programming languages used in many game engines including Unity. It is a modern, object-oriented, and type-safe programming language based on the C family of languages similar to C, C++, Java, and JavaScript. It can be used for creating secure and robust applications, with many useful features such as nullable types, exception handling, and lambda expressions to serve multiple purposes of programming.

2. Unity

   Unity is a popular free game engine used to create both 3D and 2D games for multiple platforms; in 2020, 61% of surveyed developers decided to use Unity as their preferred game engine [21], and the number increase by 31% in 2021 [22]. Unity may be described as Integrated Development Environment (IDE), providing several prominent features for creating games, such as physics, 3D rendering, collision detection, etc. Besides a simple drag-and-drop interface, Unity also provides a way to customize the game through C# coding. Compared to the similarly popular game engine, Unreal Engine [23], Unity is considered more lightweight and beginner-friendly, with various platform integrations and dedicated tools for 2D game development [24]. Apart from said features and our previous experience with this game engine, we also chose Unity due to its popularity, which makes the community pretty huge and helps when we run into trouble while developing games.

3. Visual Studio

   Visual Studio is the Integrated Development Environment (IDE) software for writing and editing code. It can work with various types of programming languages such as C#, C++, Python, etc. It is one of the most popular IDE to work with Unity since it includes many extensions and powerful features to C# programming.

4. GitHub

   GitHub is a code hosting platform for version control and collaboration through git repositories. Users can edit codes without harming the main project by creating a new branch to work. After committing changes, a branch can be merged back to the main branch by creating a pull request to let other collaborators review the code before pulling it into the master branch.

5. Figma
   Figma is a web-based app for designing and working with graphics. It has the ability to design all kinds of graphics, including websites, mobile app interfaces, prototyping designs, etc. The fact that Figma is available on multiple platforms containing various sets of designing tools available to use and collaboration features for team projects [25] makes it to be one of the most widely-used graphic tools.

6. Aseprite
   Aseprite is a pixel art tool for creating pixel art images and 2D animations. It offers various tools for creating 2D art, such as shading, pixel-perfect strokes, tiled mode, and filled and contour options. Additionally, it provides animation tools, including a real-time animation preview feature that helps users create pixel art or animation. It allows users to create multiple layers of art, separating the usage of layers for still images and frames for animations. This makes it easier to have art with multiple layers and create animations by specifying a specific layer in each frame.

2.6 Related Works

These are some works and research relating to our project, which will be our reference.

2.6.1 Similar Games Genres

Our game can be considered as various genres including visual novels, puzzles, and education.

1. Visual Novel
   A Visual Novel (VN) is a game genre that mainly focuses on story narration through text with the help of static arts, sound effects, and soundtracks, similar to a graphic novel as per the name suggested [26]. Players can interact with the game by clicking or tapping to progress the story; this system is called On-Click Progression. Furthermore, the game may provide players with extra interactions, such as dialogue choices or actions that may affect the story.

2. Puzzle Game
   A puzzle game is a problem that has a correct answer for players to discover [27]. Players must understand the concepts and features of the game and apply those rules with one's logical thinking to reach the goal. A good puzzle game should be entertaining and have an appropriate difficulty level for its target group.

3. Educational Game
   An educational game is a type of game that integrates academic and training factors into its gameplay, story narration, and other elements [28]. It requires players to use various skills such as problem-solving, strategizing, and other higher-order thinking skills in order to achieve goals and receive rewards.

2.6.2 Similar Games Example

There are many similar games to our final project, which can be used as references or ideas for other people to fortify the images of the project.

1. Dragon City
   Dragon City is a free-to-play breeding simulator game developed and published by Social Point. This game lets players build a farm of dragons on floating islands, which players can breed and collect variances of dragons to battle with other players for rewards. Each dragon can be fed and evolve into

a stronger dragon with many skills to learn. We decided to integrate their theme of collecting monsters and putting them on a farm into our game.

Due to its multiplayer-based nature, players can communicate and group up with other players as a guild to share information and passion about the game with each other, enjoying their given social fun. They can also participate in competitions between players to climb to higher ranks, fulfilling their esteem needs. There are countless dragons for players to discover and collect, giving them discovery fun.

Figure 2.17: Game Example: Dragon City
Source: Google Play

2. Princess Connect! Re:Dive

   Princess Connect! Re:Dive (Priconne) is a free-to-play fantasy RPG game developed by Cygames. Players can build teams of characters they pulled from gacha and use them to clear dungeons and battle with other players. Characters can be categorized into multiple groups based on various classification, such as standing position (front, middle, back) or roles (tanker, attacker, supporter). During battle, characters will battle automatically with their preset moves, and players can use their ultimate move when the corresponding gauge is filled. We carefully extracted some of the arena elements into our game, more details will be covered in the next chapter.

   Similar to the previous game, this game has multiplayer features like the Guild system and PVP system for players to gain esteem needs, social fun, and fellowship fun. It also has tons of refined characters to collect. Moreover, there are multiple engaging voice-over stories with animations displayed as a visual novel during the story telling section, completing players' narrative fun.

Figure 2.18: Game Example: Princess Connect! Re:Dive (Story Section)

Figure 2.19: Game Example: Princess Connect! Re:Dive (Arena Section)

3. Omega

   Omega is a puzzle game developed by ORIGIN Systems, Inc in 1989 [29]. The game lets players program tanks by using a built-in text editor with a script that is similar to BASIC. The player can learn how to write a program by using it in the gameplay to pass the game's missions and progress through the game. This game is similar to our project because the project was designed to teach the players about the genetics algorithm by incorporating it with the gameplay and game progression and make the player learn by applying the lesson through the puzzle we designed.

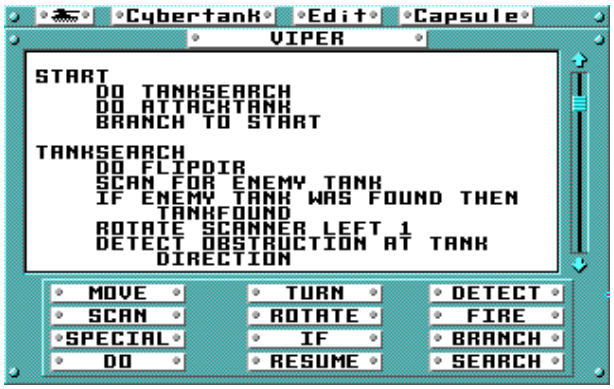

Figure 2.20: Game Example : Omega (Code Writing Section) [29]

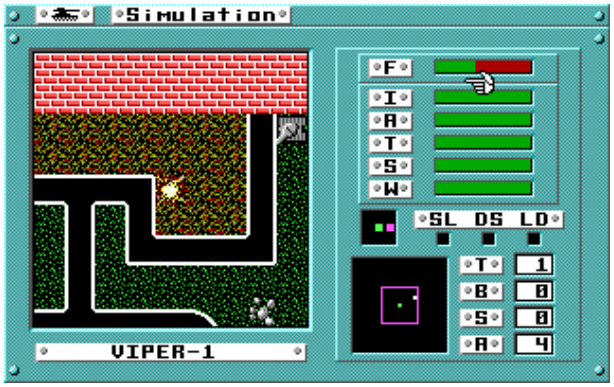

Figure 2.21: Game Example : Omega (Simulation Section) [29]

4. Super Auto Pets

   Super Auto Pets is a free-to-play auto-battler created by Team Wood Games, publicly released on Steam on 24 September 2021, before later released on mobile platforms. Players prepare their teams by buying new pets from a randomized pool and purchasing food to enhance their pets in the preparation phase as in Figure 2.22. Then they can send their pets to battle with other players in the battle phase as in Figure 2.23. These pets battle automatically, as both pets on the front row will attack each other simultaneously; if one's health point reduces to zero, it will faint and be out of combat. After the impact concludes, the most frontal pets strike again until a team, or both, has no pets left standing. Same as Princess Connect! Redive, we also extracted some of the battle features into our game.

   The game contains varieties of pets with different kinds of stats and skills for players to use in their team, letting players discover ways to synergize their pets and food the best within given limited resources, satisfying discovery fun. The game does not require players to make decisions quickly, and they can exit the game anytime during the preparation phase, thus making the game considered casual and able to satisfy players' submission fun.

Figure 2.22: Game Example: Super Auto Pets (Preparation Phase)
Source: Super Auto Pets on Steam

Figure 2.23: Game Example: Super Auto Pets (Battle Phase)
Source: Super Auto Pets on Steam

CHAPTER 3 METHODOLOGY AND DESIGN

In general, developing Digital Educational Games (DEG) has 2 main approaches. The first way is to design Computer Assisted Instruction (CAI) based on the learning topics or outcomes, then adapt it by adding the gameplay to cover the learning content to create the educational game. The second approach is reversing the first one. The game is designed first, then add a component of the learning content into the existing game to change it to be an educational game. Since we focus on the education part and already have scoped the content, we decide to develop our game using the first approach. This is one of the reasons we apply the ADDIE model in our development process.

This chapter will cover the analysis and design of both the education and gameplay aspects. The design based on the theory will be discussed. We will explain what our game contains and how it works. To clarify our idea of a design, the various diagrams and images will be also shown.

3.1 Analysis

Using the ADDIE model, the first task to do is the analysis of the problem and related detail.

3.1.1 Problem Statement

As we mentioned in the background of the project in the first chapter, we recognize that game popularity is continuously increasing, and the attention to computer knowledge should be on the same trend. But some fields of computer knowledge like the Genetic Algorithm are hard to understand. Since the game has great potential as a learning tool and there is no educational game for learning such a topic, we decided to create an educational game to help people who are interested in such a topic can learn about Genetic Algorithms simply.

3.1.2 Target Audience

People who want to learn about Genetic Algorithms, especially university students who are studying in the relevant faculties such as computer engineering. The target can be divided into 2 groups which are those who have never learned this topic before and those who have studied before but want to understand more.

3.2 Design

The main task of the designing phase focuses on defining the learning-related details which we will use the Outcome-based Education concept. Since our project is an educational game, there is also the design related to the game which will be described in the format of a game design document, for example, game overview, gameplay and mechanics, and so on.

3.2.1 Outcome-based Education (OBE) Design

Applying the OBE concept, the education module is designed according to the triangle of effective learning which starts with the designing of the learning outcomes, followed by an assessment, and the teaching and learning method.

- Learning Outcomes

  As we mentioned in the scope of the project, there are several educational topics involving genetic algorithms and real-world problems that can be solved using such algorithms.

  Using the OBE concept, we must focus on the outcome in a form of the ability that the learner should

gain [8]. So, we analyze those topics, summarize them, and figure out the core concept and skill the learner should be able to do. After consideration, we set our learning outcome using the concept of OBE together with a SMART(TT) characteristics and an action verb from the level of learning in Bloom's taxonomy [7] resulting in a single ultimate outcome which is:
The learner is able to model the algorithm for solving a real-world problem using the proper problem encoding and decoding, parent selection method, crossover method, and improvement technique including elitism and mutation.
The detailed criteria and method for measuring whether the learner achieves the outcome will be described in the next assessment session.

- Assessment
  After the learning outcome is designated, the next step is designing the assessment method. The typical way of the assessment is obviously a test in a form of an exam like the multiple-choice question or the writing exam. Since we decide to develop the educational game by designing the Computer Assisted Instruction (CAI) and adding the gameplay to it later, we create the criteria based on those typical assessing methods. For the sake of clarity and measurability, we break down the learning outcome into several criteria, grade each criterion using the Likert performance scale, and specify the condition to achieve each level in the scale using the action verb with a proper and enough detail qualifying phrase. The final assessment rubric consists of 4 criteria and at most 5 levels of performance as it is shown in Table 3.1.

Table 3.1: Rubric for Assessing the Learner

Criteria	Performance descriptors				
	Level 1	Level 2	Level 3	Level 4	Level 5
The learner is able to encode the knapsack problem as the chromosome and is able to decode it.	Able to explain the problem, constraint, and goal of the Standard, Multidimensional, and Multiple Knapsack Problem.	Able to demonstrate the chromosome encoding and decoding of the Standard and Multidimensional Knapsack Problem.	Able to demonstrate the chromosome encoding and decoding of a Multiple Knapsack Problem.	Able to solve the chromosome encoding and decoding of the Standard and Multidimensional Knapsack Problem.	Able to solve the chromosome encoding and decoding of a Multiple Knapsack Problem.
The learner is able to solve the parent chromosome selection problem in the Genetic Algorithm.	Able to explain the meaning and purpose of a parent chromosome selection in a Genetic Algorithm.	Able to demonstrate the process of the Tournament -based Selection.	Able to demonstrate the process of the Roulette Wheel Selection and Rank-based Selection.	Able to solve parent selection problems using Tournament -based Selection.	Able to solve parent selection problems using Roulette Wheel Selection and Rank-based Selection.

Table 3.1: Rubric for Assessing the Learner

Criteria	Performance descriptors				
	Level 1	Level 2	Level 3	Level 4	Level 5
The learner is able to solve the chromosome crossover problem in the Genetic Algorithm.	Able to explain the meaning and purpose of a chromosome crossover in a Genetic Algorithm.	Able to classify between Single-point Crossover, Two-point Crossover, and Uniform Crossover.	Able to demonstrate the process of the Single-point Crossover and Two-point Crossover.	Able to solve the chromosome crossover problem using the Single-point Crossover and Two-point Crossover.	
The learner is able to identify the basic method of improvement technique in the Genetic Algorithm including elitism and mutation.	Able to explain the meaning and purpose of Elitism and mutation in Genetic Algorithm.	Able to classify between Elitism, Bit Flip Mutation, Flip Bit Mutation, and Boundary Mutation.	Able to identify the type of improvement technique including Elitism, Bit Flip Mutation, Flip Bit Mutation, and Boundary Mutation.		

The assessment rubric alone cannot evaluate the learner's performance. It needs some evidence, the result of the learner's action, to pair up with the criteria for determining the actual level of the learner. We divide the type of evidence into 3 groups based on the level of learning. The assessment activity for each group is designed based on the appropriate alignment in [30]. For example, the type of the applying outcome should be evaluated using the activity causing the learner to determine what method to use and use such procedures to solve the problem. All the types of evidence we will use are listed below.

1. The evidence involving explaining, classifying, and identifying is a choice that the learner chooses for the given question or situation.
2. The evidence involving demonstrating is a result of a puzzle that they must complete with the correct processes, generating the result corresponding to the expected result.
3. The evidence involving problem-solving is a result of a puzzle similar to the demonstration, but the puzzle in this level gives more choice to the player. So, the player must decide what type of method to perform from the given description through the puzzle's element selection. Then the player should perform the method like in the demonstration.

The evidence involving explaining and demonstrating for assessment levels 1 to 3 will be collected from the test after completing the lesson in the research lab section of the game and All the types of

evidence of every level will be collected more from the result of the puzzles used for fixing the facilities when they are broken.

- Learning and Teaching
  When the learning outcome and assessment method are all designed, the final task of the education planning is to design the learning and teaching including the materials and activities. The material will be developed using multimedia mostly in the style of the image and description text.

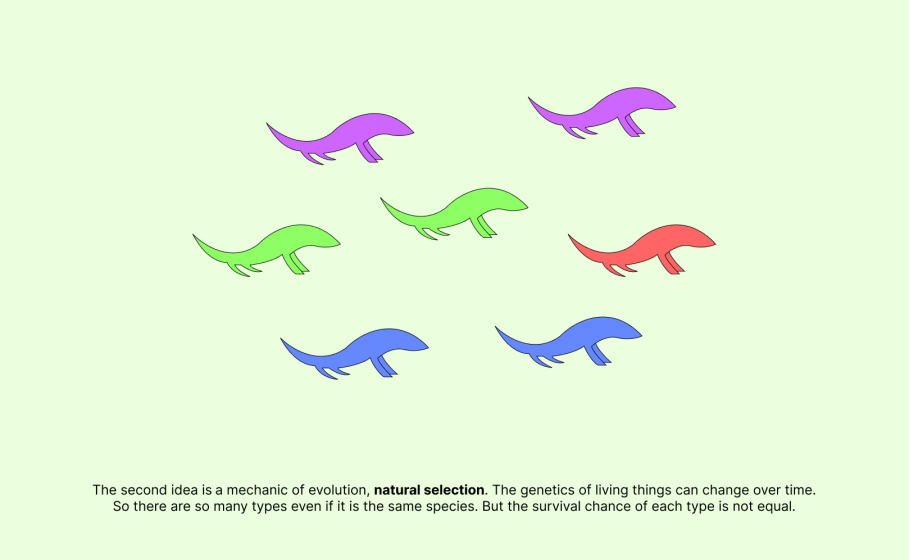

Figure 3.1: Example Material for Learning Natural Selection

Figure 3.1 is an example of the material we have already prepared for Natural Selection, one of the basic biology concepts the Genetic Algorithm is based on. The material in the figure includes the image on the center and description about natural selection at the bottom. The full learning materials are in Appendix.

When adding the component of the game, the material will be adjusted and integrated with the game story. There also will be a Research Lab system involving the material which will be described in detail in the Game Overview section.

The next issue to concern is the teaching and learning activities. Since we aim to create an educational game, we use the great advantage of using the game, the high level of interactive activities, as the learning activity in the manner of experiential learning. To be clear, the learning activity is the puzzle gameplay similar to the gameplay used for the assessment. But in the aspect of learning and teaching, there will be a guide that tells the player what and how the puzzle works. For example, the puzzle for assessing the demonstration of the player will be evaluated based on the type of process and the sequence the player chooses. But when it is used as a learning activity, the game will guide the player on what process to choose at each time step.

3.2.2 Game Design Theory Application

The game we designed mainly focuses on Mental Fun categories of Natural Funativity by Noah Falstein. We design the game to make players solve the puzzles from the lesson and manage farms and factories with money and resources in order to progress through the game.

The game we designed caters to the needs of the players following Maslow's Hierarchy of Needs to make players feel fulfilled and have answers to their needs when playing the game. The first need is Physiological

Needs where the players must obtain resources to breed stronger monsters to survive the battle inside the arena. Next is the Safety Needs where the players can improve the status of their monsters and weapons by creating new generations. For the Esteem Needs of the players we design the game to assign rank and rewards for completing quests based on the quality of the requested item to help fulfill the players when they get high rank and reward when completing the quest or win the battle in the battle arena. For the Cognitive Needs, The game was designed to teach players about genetic algorithms and how to solve the problem with this knowledge. The game's Aesthetic Needs are met through its music and visually appealing rewards with higher rankings and more powerful weapons resulting in more elaborate and appealing rewards. Lastly, the game design caters to the Self-Actualization Needs of the players by allowing them to manage their farm, factories, money, and resources according to their preferred playstyle and choose how to spend their resources and how to obtain them by themselves. The game design's hierarchy of needs is well-thought-out, providing players with an immersive and satisfying experience.

The game we design aims to provide different kinds of fun that cater to the players' preferences. The first kind of fun is Sensation, where the game's sci-fi and fantasy theme aesthetic stimulates the senses through visuals, character design, music, and sound effects from the players' interactions. The second kind of fun is Fantasy, where the game's story, world, and characters help the players feel immersed in the game and give them a role to play with objectives to accomplish in the story. The third kind of fun is Challenge, where players must complete quests and battle opponents in the arena to obtain the resources to upgrade their farms and factories. As the game progresses, the quests and opponents become more difficult, providing a sense of accomplishment upon completion. Lastly, the game design provides Expression, where the players can freely manage their money and resources, and choose the combination of weapons and monsters for the battle arena that suits their playstyle. The game design's various kinds of fun make the game engaging and enjoyable for all types of players.

3.2.3 Game Overview

As we mentioned, we decided to develop the game using an approach in which Computer Assisted Instruction (CAI) is designed first, then adding the game systems to transform it into Digital Educational Games (DEG). The main reason for adding the game systems is to increase the external learning motivation of the player so that the player has more will for playing and learning in the game. The game overview will describe how we will create the CAI and what kind of game systems we will add.

- Game Concept

  Our game will focus on three main aspects: obtaining skills, applying skills, and other activities. The player will be played as one participates in the global-guardian project, the project for creating a powerful combat force used for future fighting with the alien. To do that, the player has to research through visual novel gameplay and validate their skills using the puzzles. Then, applying their genetic algorithm skills to weapon and monster breeding. Lastly, the player can perform other activities that don't directly involve the genetic algorithm but may be related. Battles using their bred weapon and monster is an examples.

  For the first aspect, obtaining skills, there is a set of main systems involved. The player has to research via the cutscene explaining the subject. The research result (explanation for that subject) is kept in the research lab for review in the future. After that, the player is required to pass a skill check using the puzzles. And the skill-checking results are all kept in the hall of intelligence, a place a game progress is shown and can be translated into the learning outcome evaluation further.

  For the second aspect, applying skills, there is a set of primary supporting systems involved. The usual

visual novel and puzzle gameplay that focuses on education may be boredom since the learning motivation may not be enough. To drive the motivation of the player, we design supporting systems that the player should use their learned skills to create a weapon using the weapon factory and breed the monster using monster farm. To build more facilities and unlock more weapons, the player is required to complete certain research. Since the factory and farm break down after some time passes, the player must solve the same puzzle as skill validating to fix these facilities.

For the last aspect, other activities, there is a set of secondary supporting systems involved. The battle arena is a system where the player can battle with the Non-Player Character (NPC) using their bred weapon and monster. There is also a side quest where the player is required to deliver a monster with some specific appearance. Another system is a random event where a random specie of Capybara walk by facilities, and the player can capture it by rapidly clicking on it. All these systems are completely optional and don't really affect education. But they can give the player great motivation to keep playing the game.

- Game Flow Summary

  As an educational game, we conduct the game flow linearly to help the player get the basic before going into more advanced subjects. Figure 3.2 shows the game flow diagram in terms of progression and game systems. The player must achieve the condition at the arrow tail before being able to achieve the head of an arrow. For example, the player must obtain skill 1 to finish main quest 1.

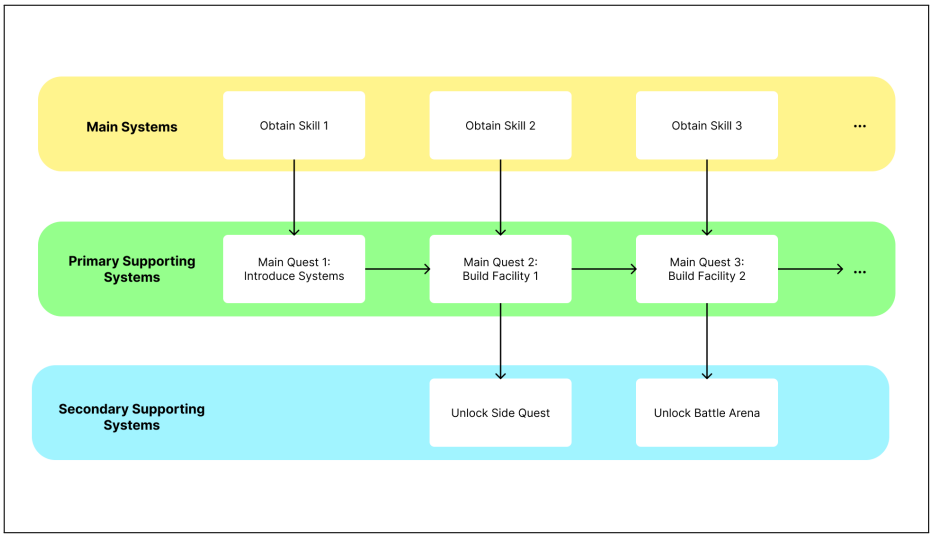

Figure 3.2: Game Flow Diagram

The game begins with the introduction to the main systems, the Research Lab and the Hall of Intelligence. After completing the early learning and getting the early skill, such as the ability to explain basic biology, the main quest will conduct the player to build more facilities that require a higher level of skill corresponding to the diagram shown in Figure 3.2. Figure 3.3 shows the main game page interface where the player can access any facility and all other systems.

Figure 3.3: Main Game Page Interface

After players unlock and build the facility, they can use that facility in the way their want. The facilities have a chance that their durability goes down after breeding. A decrease in durability results in a decrease in breeding speed and the breeding is completely stopped when all the durability run out. The player will be asked to fix the facilities through puzzle-solving. This is where the player applies their knowledge and is assessed educationally. Figure 3.4 shows an example of the interface of the puzzle the player needs to solve to fix the facility. The full gameplay of all the puzzle material can be found in Appendix.

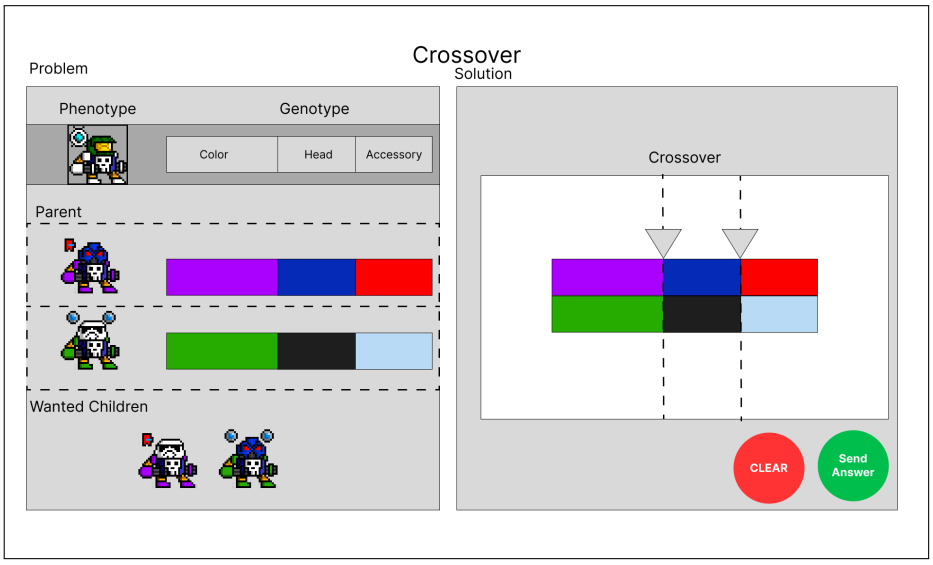

Figure 3.4: Example of Interface of the Puzzle Gameplay

Even if the main quest conducts the player continually unlocks the new facility which requires new research, the player also can enter the city freely to access other activities. Figure 3.5 shows the city interface where the player can go to the quest board and receive a side quest, buy a new monster from a shop, and battle in the arena using their own monster.

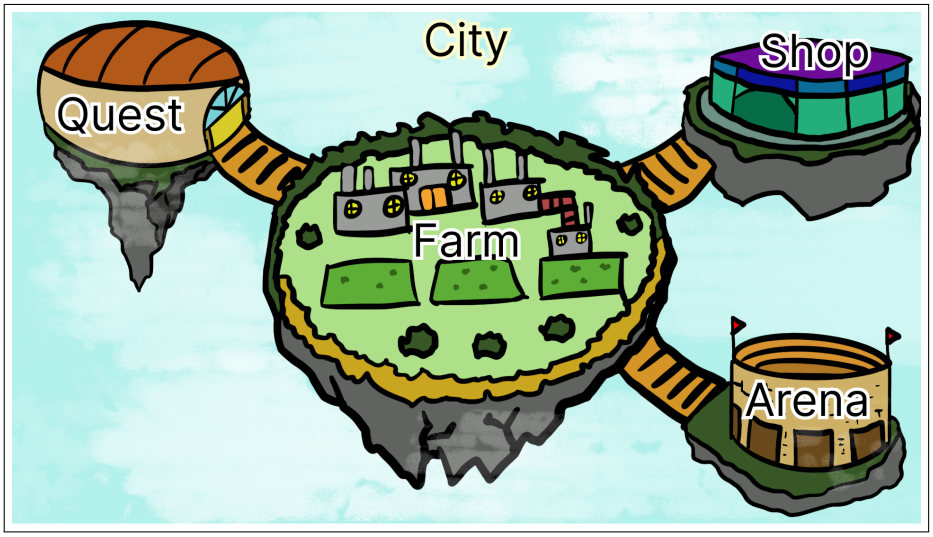

Figure 3.5: City Interface

When the game progresses to the late game, the last condition to indicate whether the player wins the game is the main quest that requires the player to unlock all the factories. Since the player is required to pass a skill-checking before unlocking a new factory, the player will be considered as completing the game as well as the learning outcomes upon completing the last main quest. And the player is allowed to continue playing in other systems freely without any restriction.

3.2.4 Gameplay and Mechanics

Within this topic, the gameplay, main mechanics, and the replaying and saving system of our game will be explained.

- Gameplay

  In this topic, we will describe how we progress the game and what the structure of both the mission and the puzzle is like.

  1. Game Progression

     In this game, players have four years to develop the most powerful weapons to defend the earth. As the game progresses, players will have to manufacture weapons in their respective factories for the government to use.

     In the beginning, players will start with a brand new company, so they have to unlock new facilities (monster farms and weapon factories) by themselves. To do that, they must satisfy specific requirements, such as learning corresponding lessons about the genetic algorithm and real-world problems, having enough resources, and passing missions.

     When players finish a lesson, they must take an exam to check their understanding of that topic, and they might have to retake that lesson if they don't pass. Players can order each facility to manufacture new generations of its corresponding product. Maintaining factories requires money, so players also have to gather those by fulfilling requests from others or battling in the arena. To finish a request, they have to submit products from their company, resulting in a reduction in the overall population, which can be solved by buying new ones from the shop.

     After the number of generations of weapons produced meets a certain number, the government will buy all the products from that factory, resetting everything. If players can submit weapons from every factory before the four years time limit reaches, they win the game.

2. Mission Structure

   There are two types of missions in this game, main quest and side quest. Main quests will help players progress the game, and side quests will provide players with money and resources to maintain their facilities.

   Players are given one main quest at a time as a guideline to tell players what they could do next, such as unlocking new facilities, learning new topics, or gaining some specific resources.

   Side quests are requests from various sources ordering products from players' companies. The reward quality depends on how the submitted product resembles the requirement of that order. If players cannot finish the request within a time limit, that side quest will fail, and players will receive a penalty in money.

- System Details

  The game systems can be described as states of the game. The states of the game and its transition are shown in Figure 3.6 below.

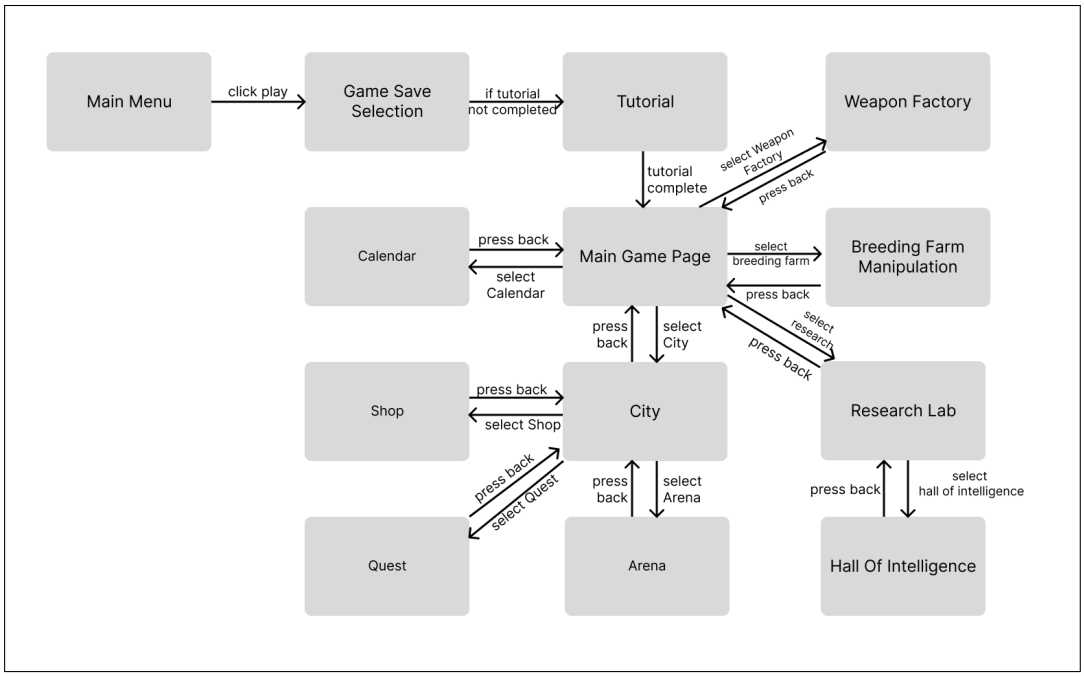

Figure 3.6: Overall Game State Diagram

– Main Systems

  These features are used for teaching and testing the players' knowledge based on the assessment criteria and level described in Table 3.1 mentioned earlier.

  1. Cutscence

     This system is designed to guide players in performing specific tasks to learn how to play the game and progress through its story. Cutscenes serve another purpose by teaching players and testing their knowledge of the lessons that were taught. Cutscenes can prompt players for answers through dialogue choices and evaluate their responses to determine whether they pass the test or not.

  2. Research Lab

     This system allows players to learn about the Genetic Algorithm and how to solve real-world problems using it. After finishing the learning on a specific subject through the cutscene, the summary of the content will be kept in the research lab for review in the future. This is a

system where players can learn about the Genetic Algorithm and how to solve Real World Problems with the Genetic Algorithm. The player must pass a requirement on a specific topic to unlock the particular customization in their breeding farm, such as a roulette wheel selection.

All the categories, learning topics, and learning content in the research lab are based on the OBE education design and will be shown in Table 3.2.

Table 3.2: The Category, Topic, and Content in Research Lab System

Category	Topic	Content
Basic Biology	On the Origin of Species	The basic concept of Evolution and Natural Selection based on Darwin's theory.
	The Genetic	The explanation of the chromosome, gene, phenotype, genotype, and biological crossover.
Genetic Algorithm	The Flow of Genetic Algorithm	The introduction to the Genetic Algorithm, the brief description of each step in the flowchart.
	Parent Selection	The process detail of the parent selection in Genetic Algorithm include Random Selection, Tournament-based Selection, Roulette Wheel Selection, and Rank-based Selection.
	Crossover	The process detail of the crossover in Genetic Algorithm include Single-point Crossover, Two-point Crossover, and Uniform Crossover.
	Improvement Techniques	The description of the improvement techniques in Genetic Algorithm include Elitism and the basic variant mutation which consist of Bit Flip Mutation, Flip Bit Mutation, and Boundary Mutation
Real-world Problems	Standard Knapsack	The problem setting, goal, chromosome encoding, and fitness evaluation of the Standard Knapsack Problem will be described.
	Multidimensional Knapsack	The problem setting, goal, chromosome encoding, and fitness evaluation of the Multidimensional Knapsack Problem will be described.
	Multiple Knapsack	The problem setting, goal, chromosome encoding, and fitness evaluation of the Multiple Knapsack Problem will be described.

3. Hall of Intelligence

   This system is an extension of the Research Lab mentioned earlier. The Hall of Intelligence provides a platform for players to test their knowledge acquired from the learning materials in the Research Lab and demonstrate their learning progress. Additionally, the Hall of Intelligence tracks players' game duration, current quest status, and the number of times they have played the puzzles or tests.

4. Assessment Methods

   This is a system of tests that players are required to pass to keep progressing through the game. The assessment methods for the players based on the performance descriptors outlined in Table 3.1 can be categorized as multiple-choice questions and puzzles. Multiple-choice questions utilize the cutscene system mentioned earlier to evaluate how well players can explain and classify their knowledge. The test consists of a series of multiple-choice questions, where

players need to choose from dialogue choices to provide their answers. The total scores are calculated based on all the answers given, and if players achieve passing marks, they successfully clear the test. However, if they do not meet the passing marks, they will fail the test. The puzzles test is designed to assess players' knowledge acquired from the lessons by allowing them to demonstrate the step from the lesson or solve the problem related to each lesson's topic. Each topic has its own platform with specific puzzles. The puzzle platform consists of Parent Selection Puzzle, Crossover Puzzle, and Knapsack Formulation Puzzle.

– Primary Supporting Systems

The systems include a monster farm and weapon factory where the player should use a portion of knowledge to play with. These systems are used to control the game progression using main quests since they are gameplay added to directly reinforce the will for learning of the player.

1. Monster Farm

   This is a Genetic Algorithm simulation for monster reproduction in farms. The individual monster will be encoded as a chromosome consisting of 2 sections of a gene as shown in Figure 3.7, each section representing each aspect of a monster including an appearance and battle status. The appearance is their head type, body color, and accessory type. The battle status includes attack, defense, health points, and speed.

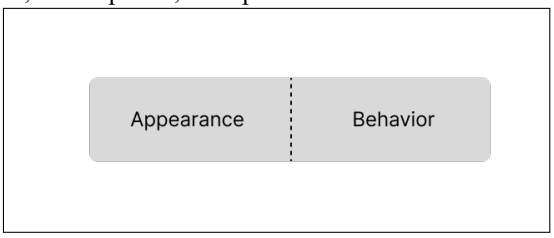

Figure 3.7: Chromosome Representation of the Monster

Players can adjust several parameters involved in the algorithm, such as the parent selection and crossover method. In addition to the basic operations of the Genetic Algorithm, players can utilize elitism, allowing them to preserve their favorite monster's chromosome for the next generation. The number of monsters on a farm and the number of breeding generations affect the amount of currency consumed which introduces the management gameplay into a game. Such currency is acquired from the main quest completion and other activities like a battle in the arena or capturing Capybara. Figure 3.8 shows the interface of the breeding farm. The players can configure the generic algorithm of the farm in the setting on the right side and set the number of generations of breeding before starting the process by paying the to-

tal money. The player can look at the monster set in the habitat or look at the monster's detail.

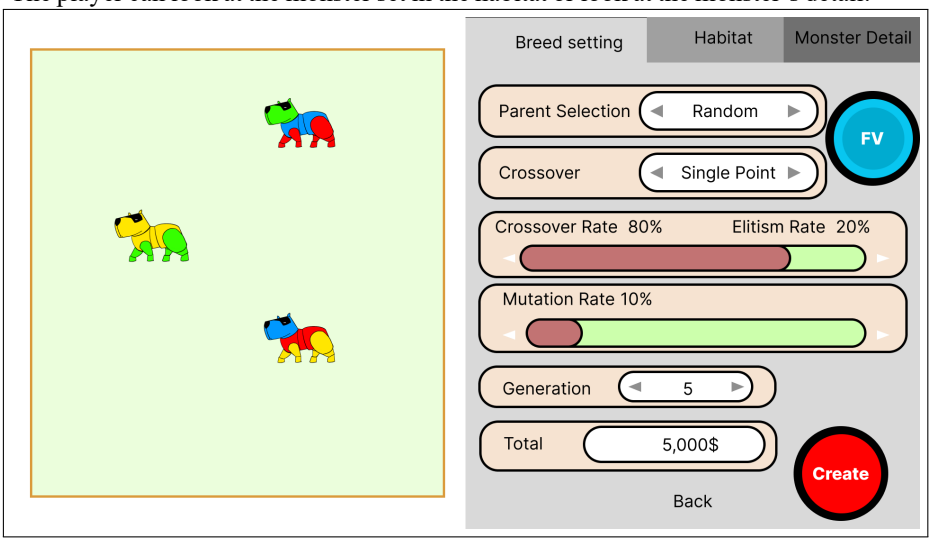

Figure 3.8: Breeding Farm Interface

2. Weapon Factory

   This system allows players to test their skills in solving real-world problems using the Genetic Algorithm. The weapons obtained from the factory are the result of attempts to solve the knapsack problem. The fitness values of each weapon's solution determine its effectiveness and appearance. These weapons can enhance the battle status of monsters, making them stronger in combat within the arena, and provide unique bonus effects specific to each factory's weapon type.

– Secondary Supporting Systems

  This is a set of systems that don't require any knowledge or skill to play with. The systems include a battle arena, side quest, and shop, which involve using their weapon and monsters in some way. The Capybara system will randomly spawn a Capybara for the player to capture. Except for the calendar system, which is used to progress the in-game activities, all the systems reward players with the resource used for breeding.

1. Battle Arena

   This game provides places for players to use their bred monsters to earn more resources in order to maintain their farms and factories; one of them is the arena system. Players can create a team of three monsters equipped with weapons from the factory to battle with enemy teams.

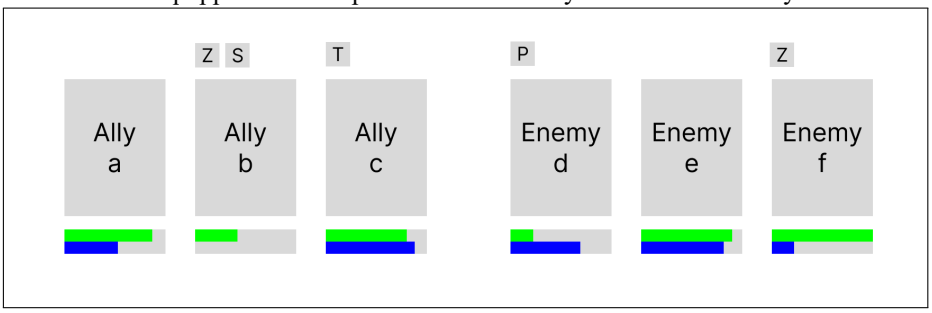

Figure 3.9: Interface Design of the Battle in the Arena

In Figure 3.9, the players' monster team is on the left and the enemy team is on the right side of the scene. Each monster will attack automatically after the speed gauge is filled (blue gauge). Attack targets will get chosen randomly depending on their corresponding space. If

the health gauge runs out, that monster is out of combat. The battle will end when the whole team is unable to battle. There is a time limit to this combat; if time runs out but the battle does not finish, the overtime phase will begin, and every monster will get a boost for a small period of time. After the overtime phase, the team with more health will win. Finally, the reward given to players of that battle depends on the result.

2. Side Quest

   Another system for players to gain resources is the side quest system. There are requests from various sources for players to fulfill by submitting monsters satisfying their requirements. The resemblance of those objects will affect the number of rewards directly. If players cannot accomplish quests they received within a time limit, those quests will fail. The designed interface of the side quest system is in Figure 3.10 and 3.11.

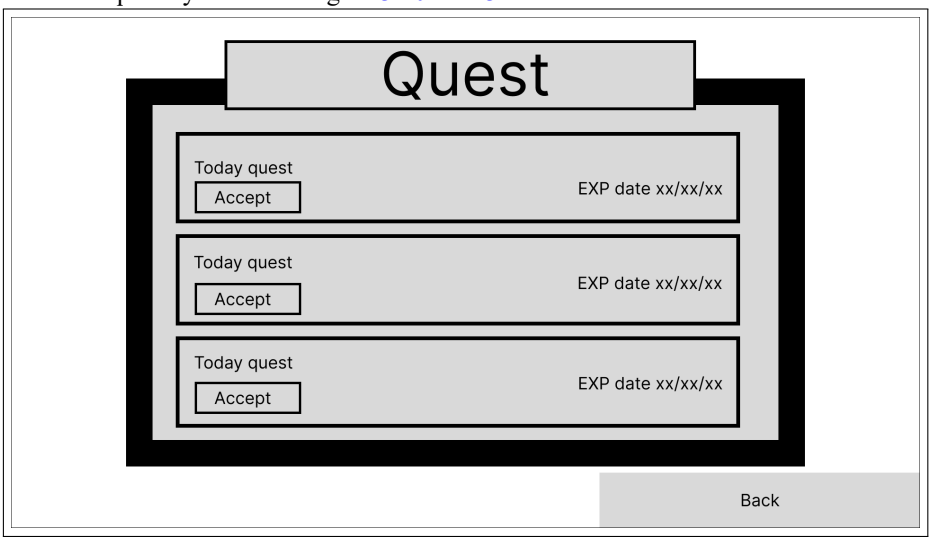

Figure 3.10: Interface of Questboard for Accept Quest Scene

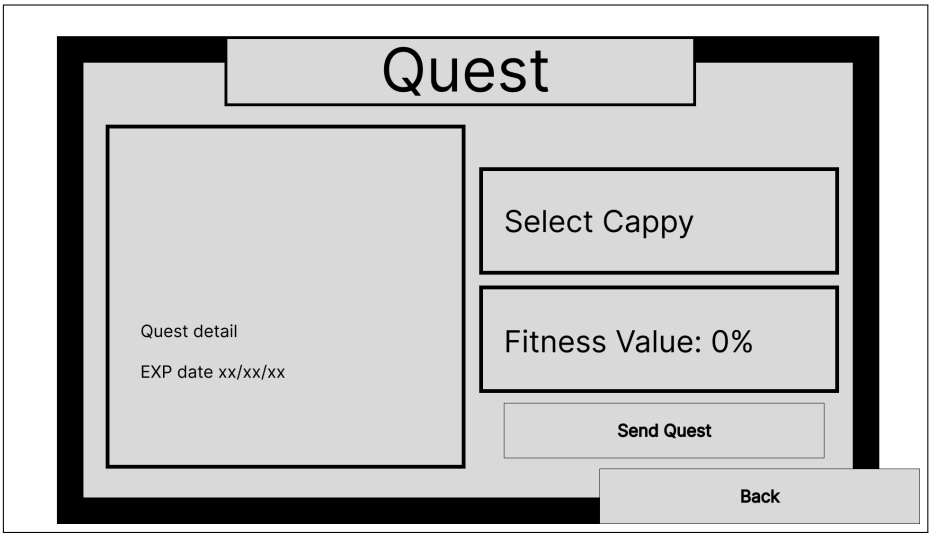

Figure 3.11: Interface of Quest Submission Scene

In Figure 3.10, the players can choose the quest they want to accept. The quest board will provide details of the quest including the appearance of the requested monster and time limit of the quest. If the players accept and in Figure 3.11, the players can submit their monster to complete the quest.

3. Shop

   After submitting quests, the population of monsters will decrease, and players could buy new ones from the shop to regain the population. The shop will refresh weekly with a new batch of randomized products for players to buy with money. The example of the designed interface of the shop page is shown as Figure 3.12.

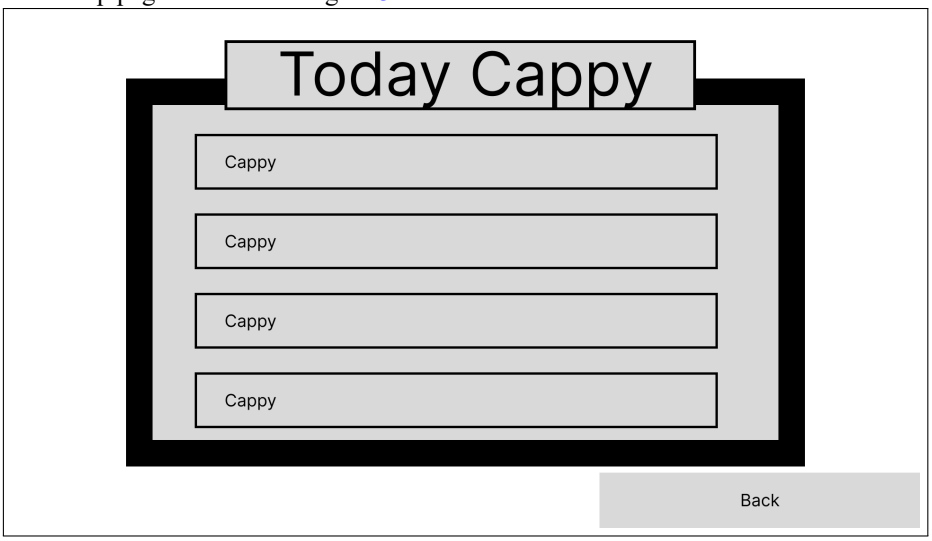

Figure 3.12: Interface of the Shop Page

4. Capybara

   This is a random event that a random species of Capybara walk by the facility on the main page. It's an addition that represents the variation in genetics using a variety in the appearance of each species. Different species hold a different number of clicks required for capture and a different amount of reward. The player can rapidly click on Capybara to capture it, and the player will get a reward upon successful capture.

5. Calendar

   Time in this game will progress when players decide to end their day. The calendar system displays many details related to date and time, such as quest submission date, breeding completion date, and other events. The game will automatically save when players end their day, and the next day will begin. The designed interface of the calendar system is in Figures 3.13.

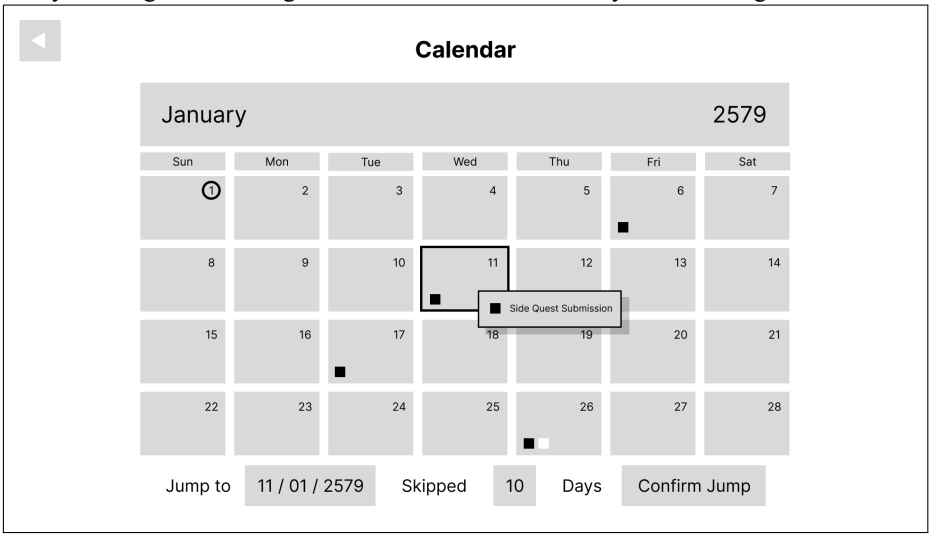

Figure 3.13: Interface of the Calendar System

- Mechanics

  Besides the mechanics described in the system detail section, the summary of the action the player can perform in our game will be described.

  1. Use Cases

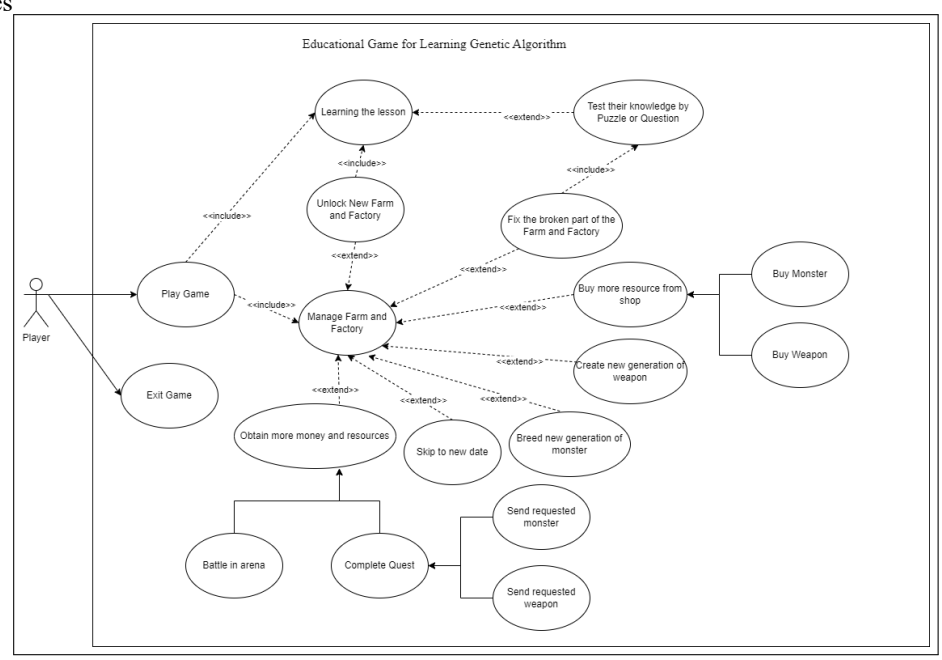

Figure 3.14: Use Cases Diagram

Figure 3.14 shows the actions the player can perform in our game.Players can start playing or exit the game when starting the program. After players choose to play the game, they can learn the lesson from the research lab or managing farms and factories in the main gameplay scene. After finishing a lesson, players must take a test to evaluate their knowledge of that topic. Also, they can unlock new facilities by fulfilling some requirements such as money, resources, and learning lessons. When farms and factories break, players have to solve puzzles using knowledge of the genetic algorithm to fix the broken part. Players can breed a new generation of monsters and weapons from corresponding farms and factories using their money or resources. They can buy more monsters or weaponry from the shop to increase the population in farms and factories. Maintaining facilities requires spending resources, so players have to manage and earn more in some way. In order to do that, they can build a team of monsters equipped with weapons to battle in the arena; or satisfy requests from people ordering monsters or weapons from players' companies.

  2. Economy

     Money is the main resource in this game. It is used in many places, including unlocking new research and facility, breeding weapons and monsters, and purchasing new monsters. Players can earn money by completing main and side quests, winning the battle in the arena, and capturing Capybara. The unlocking fee for new research is set to be very low compared to other situations to prevent the situation that money be an obstacle to learning.

- Replaying and Saving

  According to the calendar system, the game will save when players end the day and move on to the next day. Players can exit the game freely, but the game will start from the beginning of the day after the saving occurs.

3.3 Development

This phase of work aims to develop the learning materials and the learning activities. Such activities in the context of the educational game are precisely the puzzle gameplay that we will use as learning and assessment activities. The following content will describe the learning material, the structure of the puzzle, and prototyping and usability testing.

3.3.1 Learning material

From the expected outcome, we break it down and list the topic of the learning content as we mentioned in the research lab system. For each topic, the image and description text is created. Figure 3.15 is an example of the learning material we created for the single-point crossover. Please note that the material will be adapted to suit the game dialogue later, and all the learning materials are provided in the Appendix.

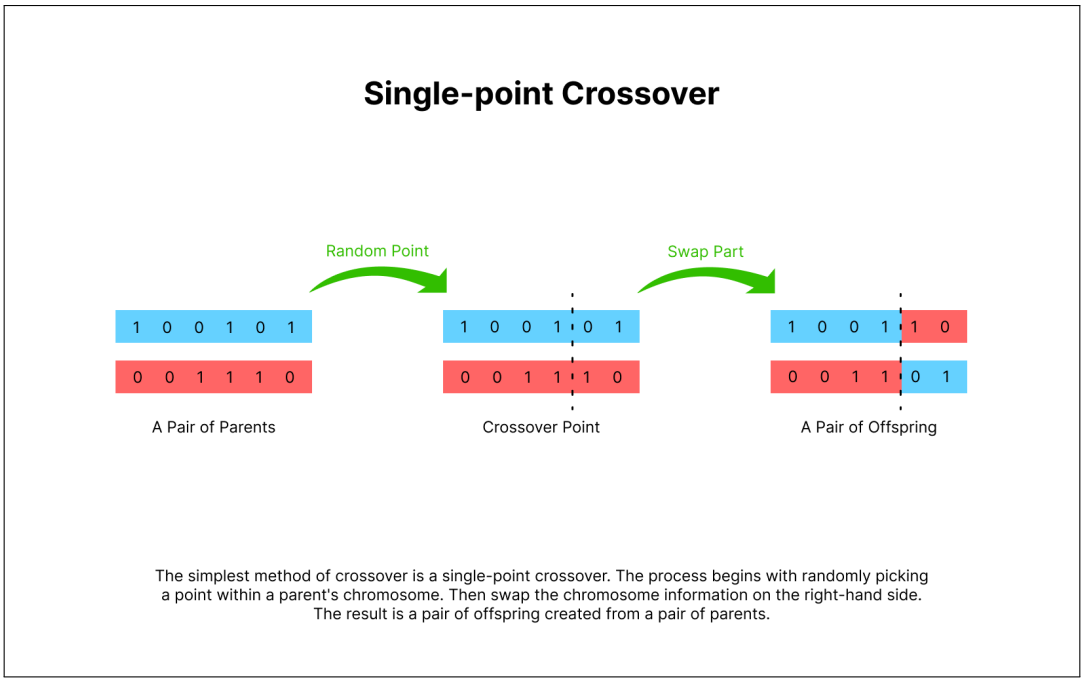

Figure 3.15: Learning Material for Single-point Crossover.

3.3.2 Puzzle structure

The puzzle will be used as an assessment activity for the demonstration and solving level of learning of the criteria in the rubric Table 3.1. These puzzles not only will be used as an assessment but also will be used as an exercise that helps the player maintain and verify their knowledge.

The players will encounter puzzles in 2 different situations. The first situation is right after the players finish learning the lesson from the research lab. The players are required to solve the puzzle to complete the lesson and unlock the new corresponding facility. This will be treated as the first assessment. Another situation is when the farms or factories break down. Players will have to solve the puzzle to fix and continue to use it. This can be interpreted as an exercise and also be calculated as a re-assessment.

The puzzle the player will encounter can be classified into 3 types as follows:

1. Questions

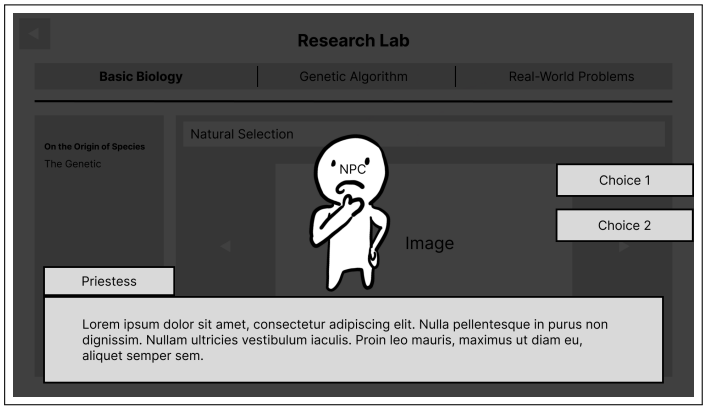

Figure 3.16: The Example of Question Scene

In Figure 3.16, the players will have to choose the answer from the question. This type of test is for testing the players on how they explain, classify, or identify the lessons they have learned.

2. Demonstration Puzzle

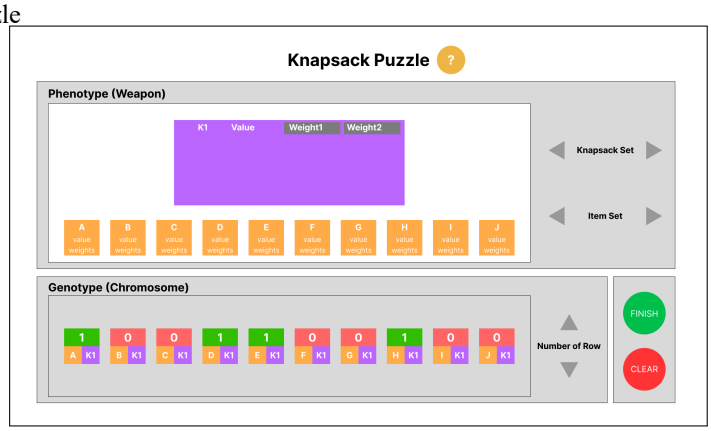

Figure 3.17: The Example of Demonstration Puzzle

This type of puzzle is an open space where the player can operate the objects on the screen to show off their skill. The result of the puzzle will be used as evidence to evaluate the players' learning level for such criteria. The example of a knapsack decoding demonstration puzzle is shown in Figure 3.17, the player is required to drag and drop the item into the knapsack above to make the phenotype match the given genotype below.

3. Problem Solving

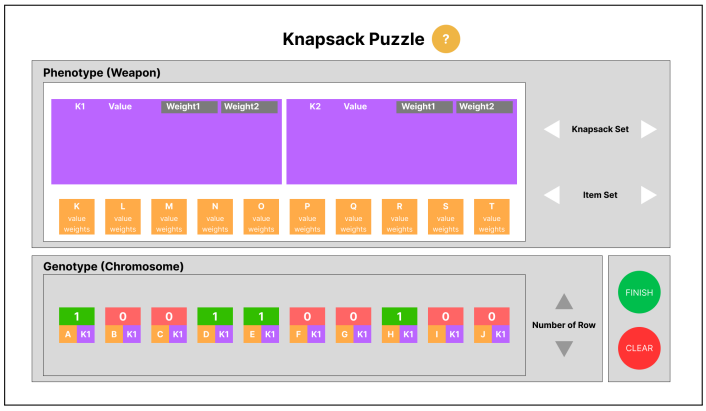

Figure 3.18: The Example of Problem Solving Puzzle

To solve the problem, the learner must choose the method of how they can solve the problem and then perform that method to solve it [30]. This type of puzzle is very similar to the demonstration puzzle with the extra step of choosing the method. The example of a knapsack solving puzzle is shown in Figure 3.18, the extra step of choosing the method is represented as a knapsack and items set selection. The player must click on the button on the right-hand side to choose the correct knapsack and item set. Then perform the drag and drop procedure just like what they must do in the demonstration puzzle to this problem-solving puzzle.

The learning and assessment activity that involves the demonstration and problem-solving level of learning includes the knapsack problem formulation, the parent selection, and the crossover. The detail of the action the player should perform in these puzzles to pass the assessment is described in Table 3.3 below.

Table 3.3: Activity to Achieve the Learning Level of each Criteria

Rubric Criteria (Topic)	Activity to Achieve Demonstration Level	Activity to Achieve Problem-solving Level
Knapsack Problem Formulation	- For problem decoding, drag and drop the item(s) into the knapsack(s) to make the phenotype match the given genotype. - For problem encoding, point and click the bitstring to make the genotype matches the given phenotype.	- For problem decoding, choose the correct item(s) and knapsack(s) set. Then perform the action like the demonstration level. - For problem encoding, choose the correct bitstring format. Then perform the action like the demonstration level.
Parent Selection	For the given parent selection method and operation buttons, point and click the operation buttons in the correct order until the number of selected parent meet the number of the current population.	Give the player the description rather than the exact name of the parent selection method with all operation buttons (unnecessary operation buttons included). The player must choose what selection method the description means and perform the action using only the necessary operation.
Crossover	For the given crossover type and the set of chromosomes, the player must pick any pair of chromosomes with the same bitstring format. Then create a random crossover point(s) to suit said crossover type, and drag and drop the proper section of chromosome to complete the crossover.	For the given set of parent chromosomes and the wanted child chromosome, the player must pick the proper pair of parents. Then select the necessary crossover point(s), and drag and drop the proper section of the chromosome to make them match the wanted child chromosome.

3.3.3 Prototyping and usability testing

For typical education design using the ADDIE model, the developed contents may be used in the education simulation to test the feasibility of the project. As an educational game, we decide to conduct usability testing instead since it's more suitable for the game and we can't directly perform the learning simulation.
Since we haven't developed the actual working software yet in this phase, we choose to develop the simple wireframe prototype, a series of mock-up screens simulating the working software, for testing purposes. Then we record the prototype as a video together with the narration of how things work.

After the prototyping is done, we decide to conduct the test on a small group of the target user which is the student that currently studies in the field of computer engineering. The data we want from this test is mainly opinions about our current design. So, we sent the video of the prototype to the tester and request them to answer the post-test question about their opinion on the design. The six questions in total with the possible answer will be shown in Table 3.4.

Table 3.4: The Questionnaire for Usability Testing

Question	Possible Answer
Have you ever studied Genetic Algorithms?	Yes/No
What do you think about this design?	Score from 1 (Terrible) to 5 (Excellent)
What do you think about the user interface?	Score from 1 (Hard to Use) to 5 (Easy to Use)
What do you think about the complexity of the gameplay?	Score from 1 (Too Complicated) to 5 (Too Simple)
If you notice any issue that makes this game terrible, what is it?	Short text
If you could change one thing in this game, what would it be and why?	Short text

CHAPTER 4 IMPLEMENTATION RESULTS

In this chapter, according to the ADDIE model, we will introduce our work in the implementation phase of our project. This includes the result of usability testing, the software implementation result, and the problem and its solution.

4.1 Usability Testing Result

We conduct usability testing using the information we already mentioned in the development phase in chapter 3. The test takes place in the first week of February 2023, which is in the first sprint of semester 2. The summary of the responses from 13 testers will be described.

Table 4.1: The Summary Response to Usability Questionnaire

Question	Response
Have you ever studied Genetic Algorithms?	8 Yes, 5 No
What do you think about this design?	An average score is equal to 4.0 (Close to excellent)
What do you think about the user interface?	An average score is equal to 4.2 (Close to easy to use)
What do you think about the complexity of the gameplay?	An average score is equal to 2.2 (Close to too complicated)
If you notice any issue that makes this game terrible, what is it?	Optional short text answer
If you could change one thing in this game, what would it be and why?	Optional short text answer

From the responses in Table 4.1 above, the overall design and the user interface have no serious problems. For the complexity of the game, the tester's overall opinion indicates that they think the gameplay is a bit too complicated. However, we think such an opinion is acceptable because our game is an educational game with puzzle gameplay which may require a portion of learning before clearly understanding the puzzle. The other 2 question is optional short-answer questions which turn out that there is some improvement suggestion and question about some specific game mechanic but still have no severe problems. So, we continue the implementation with no major change in the design. Please note that the full version of the response can be reviewed in Appendix.

4.2 Implementation Result

After we begin the software implementation for 2 Sprints, all demonstration and problem-solving puzzles that are used as learning and assessment activities are finished along with the great progression of other crucial systems related to such puzzles. The portion of the weapon factory, monster farm, main game page, and saving system is workable but not completely done. Most of the art assets also be created along with said systems.

Figure 4.1 is the screenshot of the working game scene of the knapsack problem decoding puzzle. The screenshot of the parent selection puzzle is shown in Figure 4.2. And the screenshot of the crossover puzzle also represents in Figure 4.3.

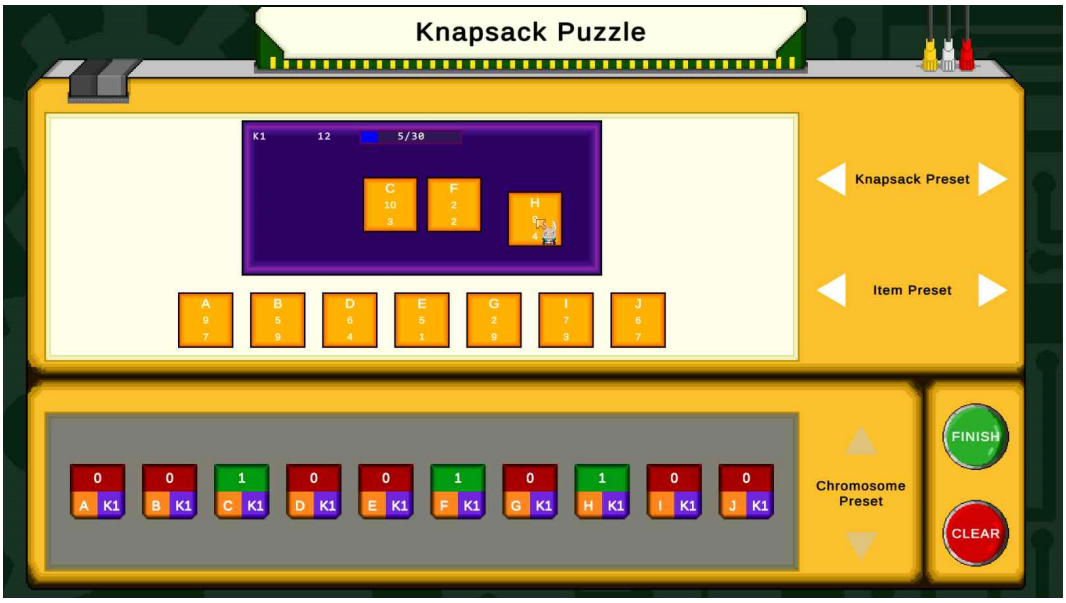

Figure 4.1: Screenshot of Knapsack Decoding Puzzle

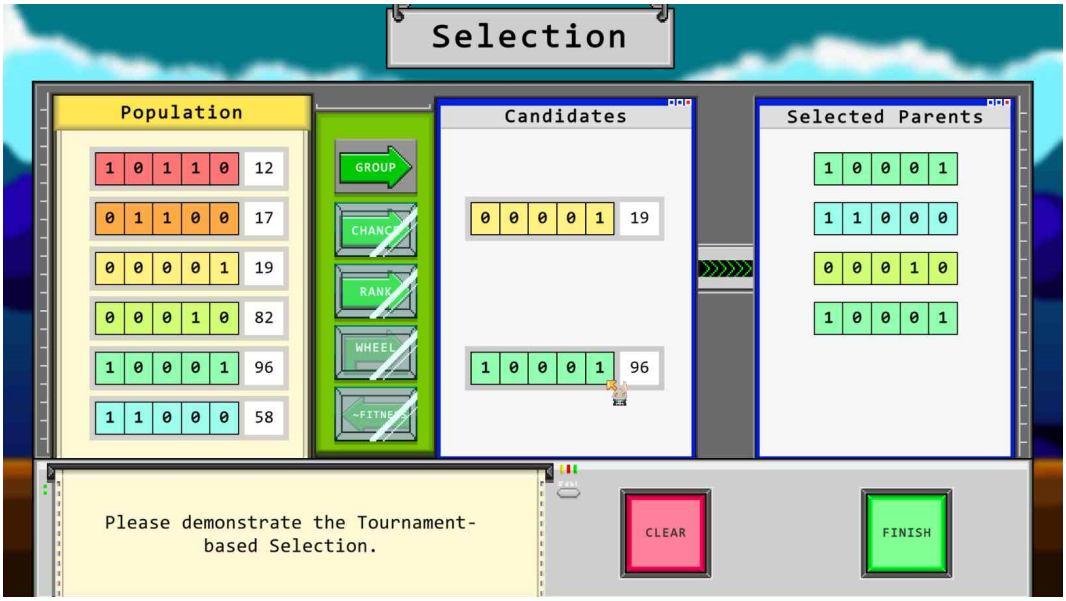

Figure 4.2: Screenshot of Tournament-based Selection Puzzle

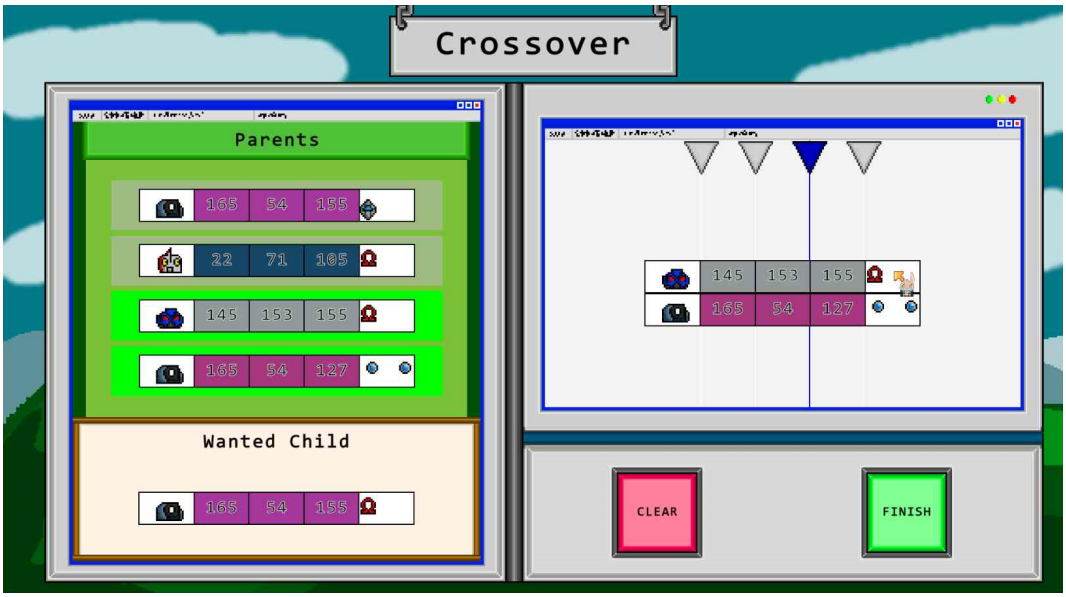

Figure 4.3: Screenshot of Single-point Crossover Puzzle

Note that both the demonstration and problem-solving levels of each puzzle type can be assessed using the same puzzle scene with some different settings. And our implemented game scene can set and check the conditions to suit the criteria we already mentioned in the development phase in chapter 3.

These screenshots below (Figure 4.4 to Figure 4.7) are from the main menu, weapon factory, mech farm, and calendar. As explained in the last chapter, the main menu will be used as a hub to access other parts of the game, such as factories and farms in the middle, city and calendar from the sides. Farms and factories will control their respective populations, from storing to breeding new generations, which players can adjust some parameters as they will. Apart from displaying the date and event within that day, the calendar will let the player end their day or skip to future days, which also results in saving the game in the process.

Figure 4.4: Screenshot of Main Menu

Figure 4.5: Screenshot of Weapon Factory

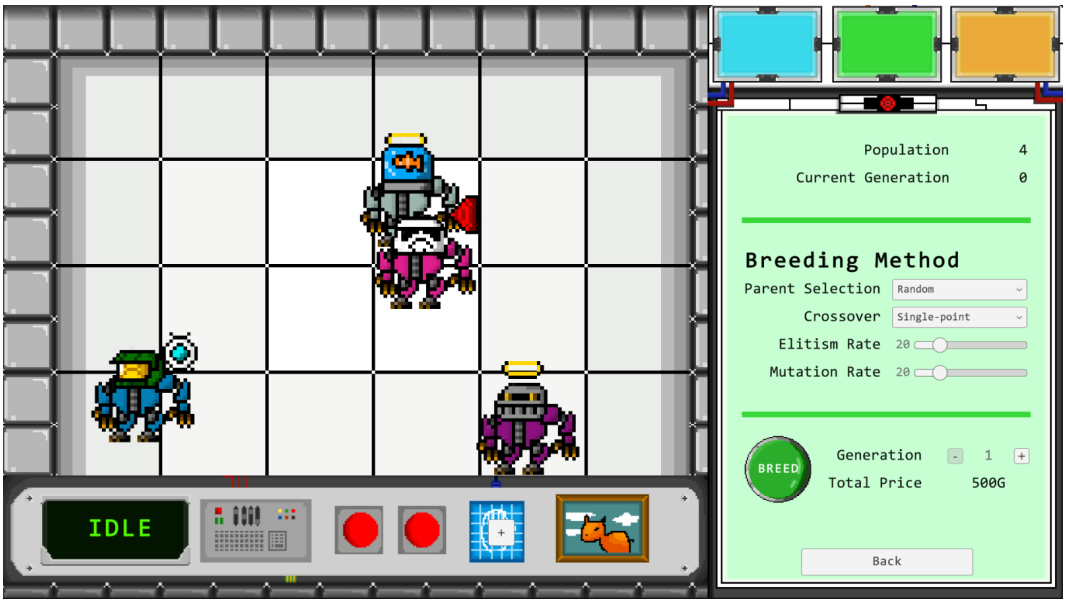

Figure 4.6: Screenshot of Mech Farm

Figure 4.7: Screenshot of Calendar

4.3 Problems and Solution

During the operation, we encountered and solved several problems. Two main problems that cause a high impact on the project are the educational game development framework and workload management. All these problems will be described in detail as follows.

1. Educational Game Development Framework Problem

   In the first two sprints, we designed the game by prioritizing educational topics over game elements via lecture-based learning; making it harder to design the gameplay and adapt those predesigned lectures into the game.

   In a lecture in the game design class (CPE467), we learned about educational game development frameworks, which are OBE and ADDIE models, so we decided to use them in our project. As a result, we could design the educational section of the game better than last time and integrate them into the game-

play more seamlessly by prioritizing which outcomes learners should have instead of what to teach them. Regardless, designing every lecture from the beginning delayed the overall design progress.

2. Workload Management Problem

   During the operation in the first semester, there were several changes in the scope of the project. We handled the changes at the very beginning of the project by revising the project proposal due to the increment of the project members. We also had to adjust and perform a vast fraction of redesigning after that because of developing framework changes. We solve these problems by continually monitoring and adjusting the remaining backlog to suit the limited time. Even if we try to redesign only the work in the necessary part and keep others the same to reduce the extra work as much as possible, we end up with extra work of around 17 man-half-day, stalling the overall progression of the project for the equivalent amount.

   After finishing three out of five sprints in the second semester, we found that there was a misestimated workload of the knapsack puzzle development, which we didn't include in the backlog in the first place. There is also an underestimated work of documentation, developing the art asset, working breeding system, and saving system since we lack experience in game development. We tackle these misestimated workloads by revising and re-allocating all the initial backlog estimations. We decide to speed up the pacing of software development, increase the initial workload estimation for the product story about documentation and adding aesthetics and decrease the workload for the story about quality assurance since we refactor a portion of the working software along with the development. A total workload of 16 man-half-day is recorded separately as extra work to reflect the authentic project progress. Figure 4.8 shows the current burndown chart of the project, which signals we might have to conduct an extra sprint if necessary.

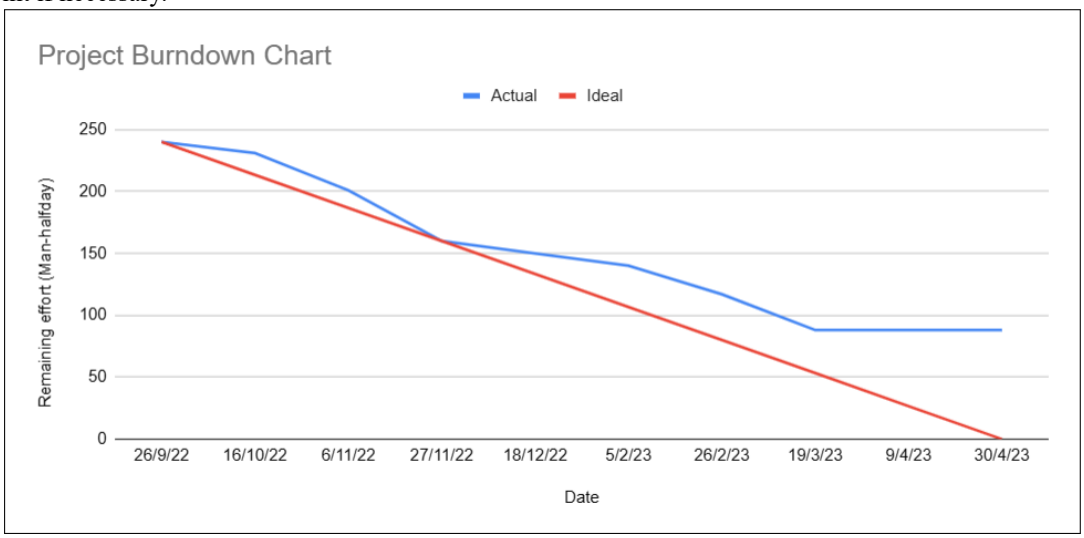

Figure 4.8: Project Burndown Chart

REFERENCES

1. Marcello A Gómez-Maureira, Max van Duijn, Carolien Rieffe, and Aske Plaat, 2022, "Academic Games-Mapping the Use of Video Games in Research Contexts," 2022.

2. Chioma Udeozor, Ryo Toyoda, Fernando Russo Abegão, and Jarka Glassey, 2022, "Digital games in engineering education: systematic review and future trends," **European Journal of Engineering Education**, pp. 1–19, 2022.

3. Jan L Plass, Bruce D Homer, and Charles K Kinzer, 2015, "Foundations of game-based learning," **Educational psychologist**, vol. 50, no. 4, pp. 258–283, 2015.

4. Steve Rabin, Ed., 2005, **Introduction to game development**, chapter 2, Hingham, MA : Charles River Media.

5. Saul McLeod, 2022, "Maslow's Hierarchy of Needs," https://www.simplypsychology.org/maslow.html#:~:text=From%20the%20bottom%20of%20the,attend%20to%20needs%20higher%20up., Last accessed 16 November 2022.

6. Chris Sniezak, 2016, "The Eight Tyeps of Fun," https://gnomestew.com/the-eight-types-of-fun/, Last accessed 16 November 2022.

7. Christine Persaud, 2021, "Bloom's Taxonomy: The Ultimate Guide," https://tophat.com/blog/blooms-taxonomy/, Last accessed 15 November 2022.

8. KMUTT C4ED, "Outcome Based Education," https://cilt.wu.ac.th/backEnd/myfile/attUKPSF/OBE%20WorkSheet%20CUPT.pdf, Last accessed 15 November 2022.

9. Regine Rafer, "CHARACTERISTICS OF GOOD LEARNING OUTCOMES," https://www.academia.edu/33889461/CHARACTERISTICS_OF_GOOD_LEARNING_OUTCOMES, Last accessed 24 November 2022.

10. ISFET, "What is the ADDIE Model?," https://www.isfet.org/pages/addie-model, Last accessed 15 November 2022.

11. Michigan State University, "PILOT TESTING IN THE SHARED DISCOVERY CURRICULUM," https://curriculum.chm.msu.edu/about/sdc-pilots, Last accessed 25 November 2022.

12. Savio D Immanuel and Udit Kr Chakraborty, 2019, "Genetic algorithm: an approach on optimization," in **2019 International Conference on Communication and Electronics Systems (ICCES)**. IEEE, 2019, pp. 701–708.

13. K.F. Man, K.S. Tang, and S. Kwong, 1996, "Genetic algorithms: concepts and applications [in engineering design]," **IEEE Transactions on Industrial Electronics**, vol. 43, no. 5, pp. 519–534, 1996.

14. Nidhi, 2017, "A Comparative Analysis of Genetic Algorithm Selection Techniques," **International Research Journal of Engineering and Technology (IRJET)**, vol. 4, pp. 2453–2455, 2017.

15. Janne Koljonen and Jarmo T Alander, 2006, "Effects of population size and relative elitism on optimization speed and reliability of genetic algorithms," in **Proceedings of the ninth Scandinavian conference on artificial intelligence (SCAI 2006)**, 2006, pp. 54–60.

16. Vinicius Fulber Garcia, 2022, "Elitism in Evolutionary Algorithms," https://www.baeldung.com/cs/elitism-in-evolutionary-algorithms, Last accessed 25 November 2022.

17. Soukaina Laabadi, Mohamed Naimi, Hassan El Amri, Boujemâa Achchab, et al., 2018, "The 0/1 multi-dimensional knapsack problem and its variants: a survey of practical models and heuristic approaches," **American Journal of Operations Research**, vol. 8, no. 05, pp. 395, 2018.

18. Bhayani Arpit, 2022, "Genetic Algorithm to solve the Knapsack Problem," https://arpitbhayani.me/blogs/genetic-knapsack, Last accessed 6 November 2022.

19. Nick Babich, 2021, "User & Usability Testing Questions: Ultimate Guide," https://xd.adobe.com/ideas/process/user-testing/usability-testing-questions-tips-examples/, Last accessed 24 March 2023.

20. Jakob Nielsen, 2012, "How Many Test Users in a Usability Study?," https://www.nngroup.com/articles/how-many-test-users/#:~:text=For%20really%20low%2Doverhead%20projects,5%20users%20per%20usability%20test., Last accessed 24 March 2023.

21. Unity, 2021, "2021 GAMING REPORT," https://images.response.unity3d.com/Web/Unity/%7B4eb56531-e6aa-492f-8fda-c68ae20af950%7D_2021_Gaming_Report_-_Operate_Solutions.pdf, Last accessed 25 November 2022.

22. Unity, 2022, "2022 GAMING REPORT," https://images.response.unity3d.com/Web/Unity/%7B10460b81-b6e7-4784-a735-e7347afdf06e%7D_Unity-Gaming-Report-2022.pdf, Last accessed 25 November 2022.

23. Perforce, 2022, "2022 GAMING DEVELOPMENT TRENDS & FORECAST," https://mma.prnewswire.com/media/1880817/Game_Development_Trends.pdf, Last accessed 25 November 2022.

24. Ben Tristem, 2022, "Unity vs. Unreal: Which Game Engine is Best For You?," https://blog.udemy.com/unity-vs-unreal-which-game-engine-is-best-for-you/, Last accessed 25 November 2022.

25. Daniel Schwarz, 2022, "Figma review," https://www.creativebloq.com/reviews/figma, Last accessed 25 November 2022.

26. Janelynn Camingue, Elin Carstensdottir, and Edward Melcer, 2021, "What is a Visual Novel?," **Proceedings of the ACM on Human-Computer Interaction**, vol. 5, pp. 1–18, 10 2021.

27. Yingfang Zhang, Zhiyu Shen, and Yifan Zhang, 2019, "Research on the Design of Puzzle Games Based on Unity 3D," in **2019 IEEE 2nd International Conference on Knowledge Innovation and Invention (ICKII)**, 2019, pp. 145–148.

28. Mary Jo Dondlinger, 2007, "Educational video game design: A review of the literature," **Journal of applied educational technology**, vol. 4, no. 1, pp. 21–31, 2007.

29. MobyGames, "Omega," https://www.mobygames.com/game/omega_, Last accessed 15 November 2022.

30. Eberly Center, 2022, "Why should assessments, learning objectives, and instructional strategies be aligned?," https://www.cmu.edu/teaching/assessment/basics/alignment.html, Last accessed 24 November 2022.