Software Engineering

Puffer Fish eats Fish Game

Problem Statement

Our inspiration for this product came from the viral video of a puffer fish eating a carrot: Pufferfish eats carrot (full video). After seeing it, we would like to create a game with this plot, which resembles games from our childhood found on websites like Friv, Miniclip about a fish in an ocean trying to survive by eating smaller fish and avoiding getting eaten. We want to add some twists to the basic game, by implementing stages and “smarter fish” which run away from bigger fish and towards smaller ones.

Some examples : https://poki.com/en/g/fish-eat-fish

Functional Decomposition

- **The player**

Moves with WASD. Its main purpose is to become as big as possible in order to not be eaten by the bigger fish in the ocean. Once it encounters a fish with a mass less or equal to its own, it is able to eat it, gaining a fraction of the object’s mass. If it accidentally comes in contact with a fish bigger than its size, the smaller fish gets eaten.

- **The NPCs**

Different in sizes, they first move in a straight line from right to left or reverse, but as the game levels up, the smaller fish run from the player and the bigger ones chase it.

- **Carrot** - the first and smallest object in the game. Once the game starts and the pufferfish is at its smallest weight, it can only eat carrots.
- **Fish** - there are 7 types of fish of different sizes. If they touch the pufferfish when it is smaller than them, they will eat it and the game is over.
- **Poisonous fish**- even though harmless in appearance, this fish will kill you if you touch it.
- **Dolphin**- the dolphin is the 5th fish by mass and the only one which is harmless. You can decide whether you eat it or not since it will not harm you.

- **Main Menu**

Implemented the Main Menu by creating a new scene and loading the Main Scene by pressing the Play button. Quit button exits the application and Settings button opens a new menu with different options:

- **Master Volume**: controlling the overall volume of the game: the background music and the sound effects
- **Background Music**: different options for the background music appear and pressing any of them changes the background music of the game. Here you can also control the volume for the background music
- **Sound Effects**: control the volume of the sound effects

- **Pause Menu**

Pause Menu can be accessed by pressing the pink fish button (Pause Menu button) in the bottom left corner. It can be accessed while playing the game and provides multiple functionalities that are enabled by pressing buttons: pause | resume | quit | main menu.

- **Game Over**

Game Over Scene is loaded when the pufferfish touches a fish that has a bigger mass and it questions the user if he wants to play again. By pressing the Yes button the Main Scene is loaded. By pressing the No button the Menu scene is loaded. From that point on the user can choose to quit the game by pressing the Quit button from the Main Menu.

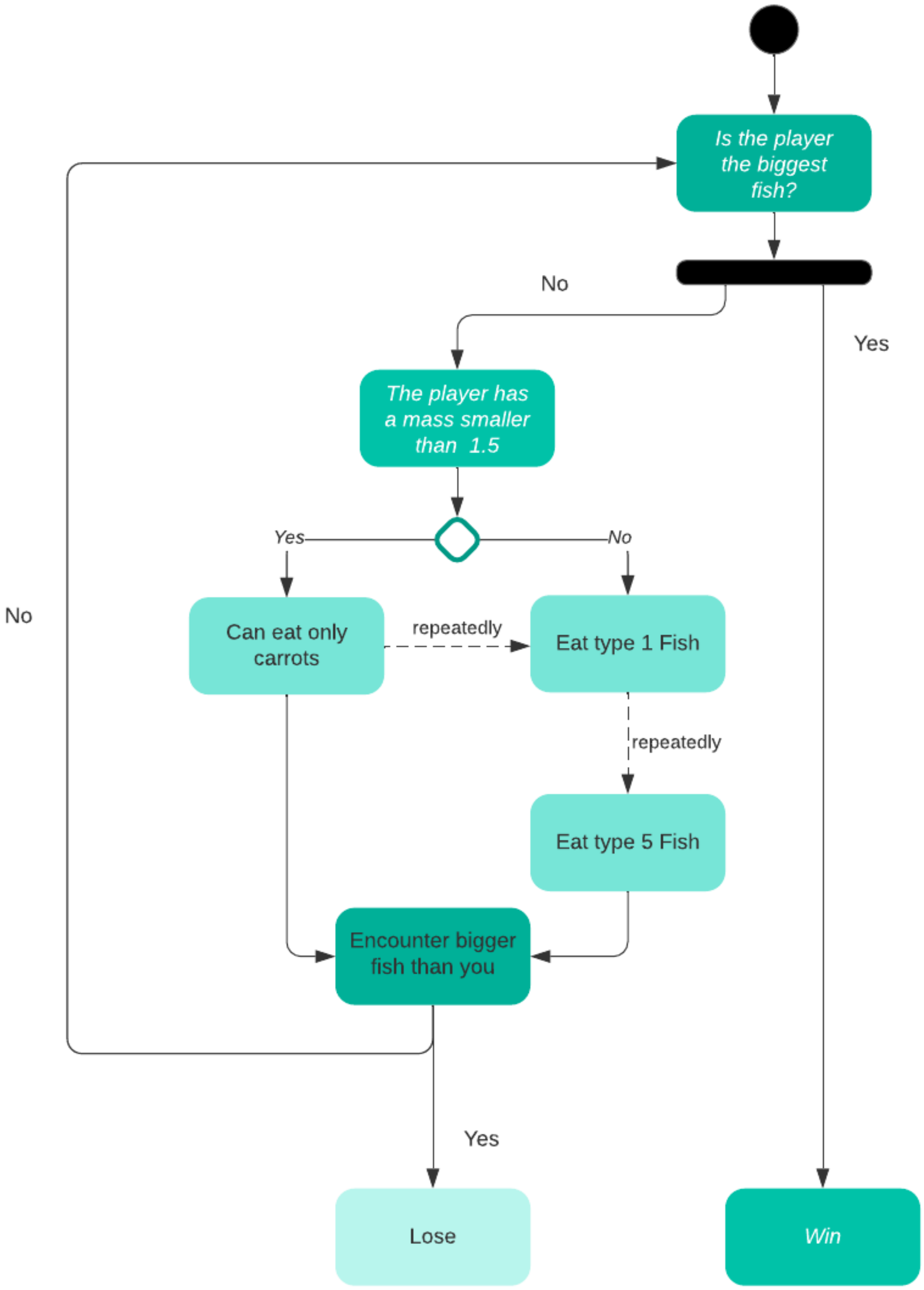
- **Winning the Game**

Win Scene is loaded when the pufferfish reaches its goal. The implementation is similar in approach with Game Over, but there is only one button displayed that takes the player back to the Main Menu by loading the Menu Scene.

Non-functional requirements list

- **Puffer fish** design, that scales according to its dimensions
- **Environment** design, that changes based on progress in the game
- **NPCs & carrots** design, that change after being eaten (NPCs turn into skeletons, carrots into scrap)
- **Main menu & settings panel** design
- **Pause menu** design
- **Background music** selection, that changes on progress in the game;
- **Sound effects** (when eating a carrot/NPC, when being eaten)
- **Game over (win, lose) splash screen** design & music

Activity/State diagram


Is the player the biggest fish?
No
Yes
The player has a mass smaller than 1.5
Yes
No
No
Can eat only carrots
repeatedly
Eat type 1 Fish
repeatedly
Eat type 5 Fish
Encounter bigger fish than you
Yes
Lose
Win

Prioritized product backlog (stories identified)

1. As a player, I want a **clear objective** of the game in order to make my strategy.
2. As a player, I want a **smooth gameplay** experience in order to enjoy the game.
3. As a player, I want to be **challenged** by the game and its enemies by increasing difficulty so that the game is not repetitive.
4. As a player, I want good and intuitive **mechanics** so I can understand how the game works fast.
5. As a player, I want **sounds and visuals** that immerse me in the game.
6. As a player, I want a **well optimized and bug free** game so that I can enjoy it.
7. As a player, I want to **see my highest score** so I can see my progress and improve at the game.
8. As a player, I want a **simple and suggestive UI** so that I can use the game with ease.
9. As a player, I want a **well optimized** game so that I can play it on multiple devices with a good quality.
10. As a developer, I want **good task tracking** among team members so that everyone can communicate easily with each other about the things they are working on.
11. As a developer, I want **regular updates** about the game's progress so that we can come up with new ideas and solutions together.
12. As a developer, I want **constant game testing** so we can prevent bugs or bit errors.

Project charter document

Project objectives

- Player, NPCs, background assets
- Music, sound effects samples
- Game start, game over, win - scenes
- Environment
- Player movement
- Object and border collisions
- NPCs basic movement
- NPCs smart movement
- Carrot movement
- NPCs and carrots spawners
- Special NPCs
- Menu creation
- Menu options + design
- Eating mechanics (carrots, other fish, the player)
- Gaining mass mechanic + growing fish
- Bigger fish decreased speed
- Fish skeletons and carrot scraps falling after being eaten
- Stage one
- Stage two
- Stage three

Vision

Our vision targets people from all around the world and their need for entertainment. We want to build an application that keeps them engaged.

Scope

To fulfill our vision, we would like to bring out a game that combines a good meme with the nostalgia of games we used to play in our younger days.

Team & responsibilities

- Team Leader - **Andreea Ciurescu**
- Developers - **David Bejenariu, Miruna Vasiliu**

Stakeholders

We really thought about people our age or younger, who used to play this kind of games or still do, but really anyone may find our game enjoyable, especially people knowing the Puffer fish meme, and would count as stakeholders. Releasing the game for free would also make it more accessible.

Budget

We are currently students and cannot afford premium hosting, training, development or marketing. The game is made in Unity - which is a free development tool. Thus, our budget would consist of **100$**, to cover its release to the Steam platform. We plan to make it free and introduce in-app purchases, such as skins/chromas for the puffer fish.

Roadmap

Sprint 1 (29 nov - 12 dec 2022)

- Player, NPC’s, background assets
- Music, sound effects samples
- Environment (borders)
- Player movement
- NPC’s basic movement
- Carrot movement
- Object collisions

Sprint 2 (13 dec - 26 dec 2022)

- Menu options + design
- Menu creation
- Eating mechanics (carrots, other fish, the player)
- Special NPC’s
- Game start, game over, win - scenes
- NPC’s and carrots spawners
- Stage one

Sprint 3 (10 jan - 23 jan 2023)

- Stage two
- NPCs smart movement
- Bigger fish slower speeds
- Stage three
- Gaining mass mechanic + growing fish
- Fish skeleton falling after eating
- Final Bug fixes

Definition of Done and Definition of Ready

Definition of Ready:

- All user stories meet the **INVEST** criteria:
  - I - Independent
  - N - Negotiable
  - V - Valuable
  - E - Estimable
  - S - Small
  - T - Testable
- All user stories and acceptance criteria are clearly defined
- User stories are refined by the development team and stories are estimated
- Point of contact for questions is defined
- Non-functional requirements are defined

Definition of Done:

- Acceptance criteria is met
- Unit/integration tests pass
- Code reviewed by all team members
- Testing done on multiple devices
- Functionality is documented if necessary
- Documentation is updated