Serotope Software Design Document

Benedict Hobart Leanne Capewell Jordan Burdett

|  |
| --- |
| Table of Contents  1. **Introduction**    1. Purpose    2. Scope    3. Overview    4. Definitions and Acronyms 2. **Use Cases**    1. Actors    2. List of Use Cases    3. Use Case Diagrams    4. Use Cases 3. **System Overview**    1. Component Diagram    2. Domain Model    3. System Interfaces 4. **Design Overview**    1. Classes    2. Class Dictionary 5. **Dynamic Model**    1. Sequence Diagrams    2. State Diagrams 6. **Data Design**    1. Data Description    2. Data Dictionary 7. **Human Interface Design**    1. Overview of Human Interface    2. Screen Images    3. Screen Objectives and Actions 8. **Appendices** |

# 1.Introduction

This software design document describes the architecture and system design of Serotope to aid in its development by detailing how it should be built. Within this document are narratives and semi-formal notation detailing its design including use case models, sequence diagrams, domain models and class diagrams.

## 1.1.Purpose

The purpose of this design document is too provide the developers with a astute understanding of what is to be done in response to the objectives of our stakeholders it does this by providing a description of the software system to be built.

## 1.2.Scope

This software design document details the design for a educational game ‘Serotope” that provides students with a means to understand the principles of mendelian inheritance in a intuitive sense. it is not meant to be used as the sole method of education for mendelian inheritance.

## 1.3.Overview

Serotope is a multi-directional shooter similar to the game ‘Asteroids’, the user controls a creature by moving it around a two-dimensional world inhabited by other creatures. The objective of the game is to get the high-score by surviving the longest amount of time, the user does this by moving, shooting and ‘reproducing’ with other creatures to produce offspring and thus survive for another generation.

## 1.4.Definitions and Acronyms

* **JBOX2d *– a Java port of the physics engine Box2d initially designed for C++***
* **Slick *– a java based game engine built on the LWJGL framework***
* **Add more as we add more to the design document**

# 2.Use Cases

## 2.1.Actors

### 2.1.1 User

The User is the person who interacts with the game, this is most likely a student playing the game for education but this abstraction represents that all users will perform similar actions.

### 2.1.2 Database

The database is the local data saved on the Users computer, this is how the system under design manages highscores and achievements, with possible implementation of save-states

### 2.1.3 System Under design

The system under design is the educational game and its interactions with its component subsystems that is being created. This actor represents the system and the actions it takes.

## 2.2.List of Use Cases

### Primary Use Case

2.2.1 Navigate Menu

2.2.2.Start Game

2.2.3.Play Game

2.2.4.Move creature

2.2.5.Creature attack

2.2.6.Creature Death

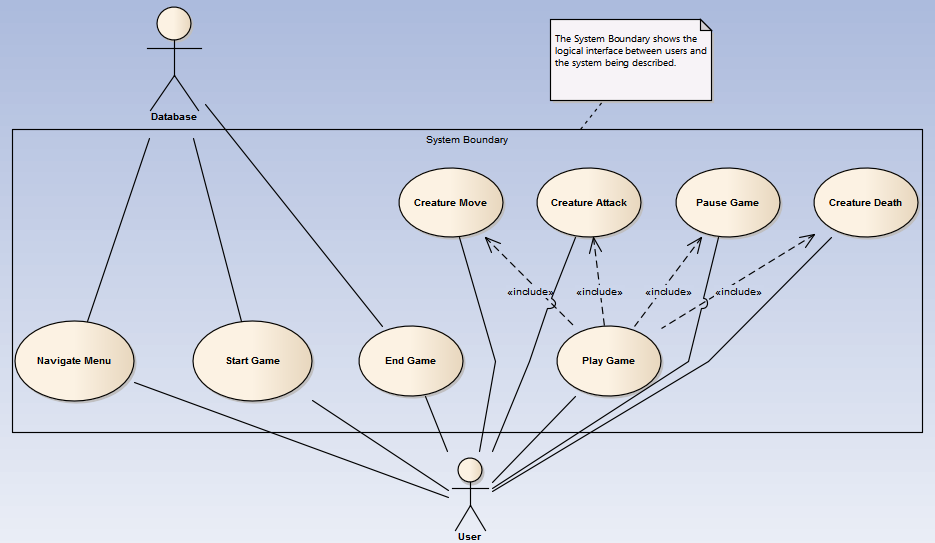
2.2.7.Pick up DNA

2.2.8.Pause Game

2.2.9.End game

## 2.3.Use Case Diagram

### Primary use case diagram



## 2.4.Use Cases

|  |
| --- |
| 2.4.1 Navigate Menu **Actors**  User  Database  **Stakeholders and their needs**  User – wants to explore the game menu, enabling them to choose options such as starting a game, viewing high scores, or exiting the game  **Preconditions**  User has loaded the game and is at the main menu screen.  **Postconditions**  None  **Trigger**  User opens the game  **Basic flow**   1. Main menu screen is displayed 2. User selects an option (Play, Achievements, Settings) 3. Chosen option is executed   Extensions  2a. User selects 'Achievements'  2a1. Local storage checked for game data  2a1a. Data is present  2a1a1. Load data  2a1b. Data isn't present  2a1b1. Create local store |

|  |
| --- |
| 2.4.2.Start Game **Actors**  User  Database  **Stakeholders and their needs**  User – wants to start the game so they can play  **Preconditions**  None  **Postconditions**  None  **Trigger**  User has selected “play” on the main menu  **Basic flow**   1. a black loading screen is displayed and control is taken away from the User 2. Assets are loaded 3. Once assets have loaded the game screen fades in and control is passed back to the User 4. After game has loaded refer to use Case 3 (Play Game)   **Extensions**  2a. Assets not present  2a1 Fatal error. Tell user to reload the game  2b game is out of focus  2b1 pause game |

|  |
| --- |
| 2.4.3.Play Game **Actors**  User  Database  **Stakeholders and their needs**  User – wants to enjoy himself  **Preconditions**  None  **Postconditions**  None  **Trigger**  User has started a game.  **Basic flow**   1. The User creature is 'Born' in the centre of the screen 2. The game loop proceeds until the game 'ends' 3. The game ends, refer to Use Case 9 (End Game)   **Extensions**  – Alternative action flows  – May begin and end anywhere  Extensions may have extensions |

|  |
| --- |
| 2.4.4.Move creature **Actors**  User  Database  **Stakeholders and their needs**  User – wants to move their creature on the screen  **Preconditions**  User has started a game and it is currently being displayed on the screen  **Postconditions**  None  **Trigger**  User presses one of the move buttons on the keyboard (default is arrow keys)  **Basic flow**   1. User's creature moves in the direction indicated by the user's input (up, down left or right). 2. Movement speed and acceleration is based on the creatures 'traits'.   **Extensions**  1a. User pressed more than one button  1a1. The user's creature moves in a direction that is the average of all the pressed directions.  1b Another creature is in the area the user is trying to move to  1b1. The user's creature is deflected around the other creature. |

|  |
| --- |
| 2.4.5.Creature attack **Actors**  User  Database  **Stakeholders and their needs**  User – Wants to attack other creatures by shooting at them  **Preconditions**  User has started a game and it is currently being displayed on the screen  **Postconditions**  None  **Trigger**  User presses one of the shoot buttons on the keyboard (default is W, A, S or D)  **Basic flow**   1. User presses the shoot button on the keyboard 2. Bullets are displayed on the screen, moving in the chosen direction, away from the creature. The number of bullets and bullet speed is determined based on the creature's 'traits' 3. The bullet is removed from the game when it interacts with another creature or moves too far off the screen.   **Extensions**  4a Bullet collides with another creature  4a1 That creature takes damage, reducing its hit points based on the strength of the bullet. |

|  |
| --- |
| 2.4.6.Creature Death **Actors**  User  Database  **Stakeholders and their needs**  User – Wants to kill other creatures and not die  **Preconditions**  User has started a game and it is currently being displayed on the screen  **Postconditions**  None  **Trigger**  A creature has health less than or equal to zero.  **Basic flow**   1. Creature death animation is displayed. 2. Creature is de-spawned. 3. The creature's 'DNA' is left in the position where the creature died.   **Extensions**  2a Creature that died was controlled by the user.  2a1 The game ends (refer to 'End Game' use case) |

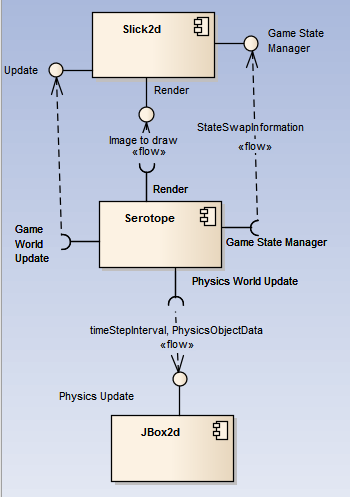
|  |
| --- |
| 2.4.7.Pick up DNA **Actors**  User  Database  **Stakeholders and their needs**  User – Wants to evolve their creature into one with better traits.  **Preconditions**  User has started a game and it is currently being displayed on the screen  **Postconditions**  None  **Trigger**  User's creature moves to the location of a piece of DNA  **Basic flow**   1. User's creature is displayed at same location as the piece of 'DNA'. 2. The 'DNA' is no longer displayed. 3. User is no longer in control of their creature. 4. Old player controlled creature is taken over by AI 5. A new creature spawns with the player controlling it   **Extensions** |

|  |
| --- |
| 2.4.8.Pause Game **Actors**  User  Database  **Stakeholders and their needs**  User – Wants to pause the game so they can resume it later.  **Preconditions**  User has started a game and it is currently being displayed on the screen  **Postconditions**  **Trigger**  User presses the pause button  **Basic flow**   1. The game state is paused. 2. All creatures and projectiles stop moving. 3. Timers stop 4. An pause overlay screen is displayed. 5. User chooses to resume game. 6. The game state is restored to exactly the same as it was when it was paused.   **Extensions** |

|  |
| --- |
| 2.4.9.End game **Actors**  User  Database  **Stakeholders and their needs**  User – wants to stop playing the game  **Preconditions**  User has started a game and it is currently being displayed on the screen  **Postconditions**  High score screen is displayed.  **Trigger**  User's creature's health is less than or equal to zero.  **Basic flow**   1. Creature death animation is displayed 2. ‘Game Over' appears on the screen 3. User's score appears, along with the high score list   **Extensions** |

# 3.SystemOverview

## 3.1.Component Diagram



## 3.2.Domain Model

## 3.3.System interfaces

# 4.Design Overview

## 4.1.Class diagram

## 4.2.Class dictionary

# 5.DynamicModel

## 5.1.Sequence Diagrams

## 5.2.State Diagrams

# 6.Data Design

## 6.1.Data Description

## 6.2.Data Dictionary

# 7.Human Interface Design

## 7.1.Overview of Human Interface

## 7.2.Screen Images

## 7.3.Screen Objectives and Actions

# 8.Appendices