**Software Requirements and Design Document**

**For**

**Group 2**

Version 1.0

**Authors**:

Ricardo Jimenez

Austin Leach

Bradford Greene

TiRon Anderson

Marc Montero

# Overview (5 points)

Jurassic Expedition (JE) will be a two-dimensional adventure game. The objective is to retrieve a dinosaur egg to take back to the present and repairing your time machine to make the trip back. The environment will be “open-world”, so the player may move around as they please amongst unlocked areas. Obstacles will be overcome by either defeating an enemy in combat, or solving a puzzle. JE will progress in an overall linear fashion, meaning the player will be required to complete one task before moving on to the next.

We have two potential ways to add on to our game, if rate of progress permits. The first would be expanding on the puzzles by either adding more or making the existing ones more complex. The second is making the game more dynamic and dependent on players choices.

# Functional Requirements (10 points)

High

* Controls – moving character and navigating menus
* Combat system
* Characters objects – How interactions will trigger events.

Medium

* Level design
* Artwork
* Sound effects
* Texts boxes
* Puzzle designs and implementation

Low

* More advanced puzzles. Prioritizing establishing simple ones first.
* Item storage – more of an inessential cosmetic factor

# Non-functional Requirements (10 points)

Performance: can load scenes in 1 second. Should have negligible input lag (action should execute within 0.25 seconds of player entering input).

# Use Case Diagram (10 points)

Diagram

Description automatically generated

|  |  |
| --- | --- |
| **UML Use Case Specification** | |
| **Unique Name** | Play Game |
| **Brief Description** | Describes how a user will play the game |
| **Participating Actors** | Player |
| **Preconditions** | The player’s device meets all the software requirements. |
| **Postconditions** | The game is successfully exited, and user is returned to desktop. |
| **Basic Flow of Events** | 1. The player will hit the “play” button on the main menu 2. The player will navigate the map 3. The player will interact with the baby dinosaur 4. The player will continue to navigate the rest of the map 5. The player will enter combat with the triceratops 6. The player will navigate the map 7. The player will enter combat with the pterodactyl 8. The player will navigate the map 9. The player will enter combat with the tyrannosaurus rex 10. The player will navigate the map 11. The player will enter the cave 12. The player will navigate the cave 13. The player will interact with all levers in the cave 14. The player will navigate the cave 15. The player will enter the maze 16. The player will navigate the maze 17. The player will find the toolbox 18. The player will win the game 19. The player will end the game |
| **Alternative Flows** | * Collect coins in step 16   + The player will collect coins as they navigate the maze   + The game then continues to step 17. |
| **Special Requirements** | N/A |

|  |  |
| --- | --- |
| **UML Use Case Specification** | |
| **Unique Name** | View Controls |
| **Brief Description** | Describes how a player will view the controls of the game. |
| **Participating Actors** | Player |
| **Preconditions** | The player’s device meets all the software requirements. |
| **Postconditions** | The player will be back at the main menu of the game. |
| **Basic Flow of Events** | 1. The player will click the “how to play” button on the main menu 2. A text box with the controls will display on the screen 3. The player will click the back button |
| **Alternative Flows** | N/A |
| **Special Requirements** | N/A |

# Class Diagram and/or Sequence Diagrams (15 points)

Player class

* Attributes:
  + Sprite
  + Storage - an array of Item class pointers
* Operations:
  + Movement
  + Interaction – triggers flags to update environment and open menus.

Partner class

* Attributes:
  + Sprite
  + Stats – Health, attack options
* Operations:
  + Movement

Dinosaur class

* Attributes:
  + Sprite
  + Stats – Health, attack options
* Operations:

Item class

* Attributes:
  + Sprite
  + Name
* Operations:
  + None

Environment Class

* Attributes:
  + Sprite
* Operations:
  + Reactive – responds and updates when player object interacts with it.

Puzzle class

* Attributes:
  + Display
* Operations:
  + Reactive – responds to input from player

Combat class

* Attributes:
  + User interface
* Operations:
  + Takes Partner object and one Dinosaur object and loads attributes
  + Reads user input and updates
  + Execute battle animations.

# Operating Environment (5 points)

Our game will work on a PC that uses a version of MacOS, Windows, or Linux that can run Unity version 2020.1.7f1. It does not interact with other applications or multiple devices.

# Assumptions and Dependencies (5 points)

There are no dependencies since it has no interaction with other applications or devices.