**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)

*Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).*

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 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)

Our project does not feature any non-functional requirements.

# Use Case Diagram (10 points)

*This section presents the* ***use case diagram*** *and the* ***textual descriptions*** *of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

***Textual descriptions of use cases****: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*

Player case

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

*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

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:

Dinosaur class

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

Item class

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

Environment Class

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

Puzzle class

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

# 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.