|  |
| --- |
| Technical Design Document |
| Dinosaur Simulator |
| Richard Delamore |

|  |
| --- |
|  |

Contents

[1.0 Revision History Version 2](#_Toc11750920)

[2.0 Development Environment 2](#_Toc11750921)

[2.1 Game Engine 2](#_Toc11750922)

[2.2 IDE 2](#_Toc11750923)

[2.3 Source Control procedures 2](#_Toc11750924)

[2.4 Third Party Libraries 2](#_Toc11750925)

[2.5 Other Software 2](#_Toc11750926)

[3.0 Game Overview 2](#_Toc11750927)

[3.1 Technical Goals 2](#_Toc11750928)

[3.2 Game Objects and Logic 2](#_Toc11750929)

[3.3 Game Flow 2](#_Toc11750930)

[4.0 Mechanics 3](#_Toc11750931)

[5.0 Graphics 4](#_Toc11750932)

[6.0 Artificial Intelligence 4](#_Toc11750933)

[7.0 Physics 6](#_Toc11750934)

[8.0 Items Item 7](#_Toc11750935)

[9.0 Game Flow 7](#_Toc11750936)

[9.1 ‘Mission’ / ‘Level’ structure **Error! Bookmark not defined.**](#_Toc11750937)

[9.2 Objectives **Error! Bookmark not defined.**](#_Toc11750938)

[10.0 Levels 7](#_Toc11750939)

[11.0 Interface 8](#_Toc11750940)

[11.1 Menu 8](#_Toc11750941)

[11.2 Camera 8](#_Toc11750942)

[11.3 Controls 8](#_Toc11750943)

[12.0 Asset List 8](#_Toc11750944)

[13.0 Technical Risks 8](#_Toc11750945)

# Revision History Version

|  |  |
| --- | --- |
| Version | Description |
| 1.0 | Initial Document |
| 1.1 | Update the section on the Dijkstra’s Algorithm |

# Development Environment

## Game Engine

* Bootstrap

## IDE

* Visual Studio.

## Source Control procedures

* Github

## Third Party Libraries

* CRT Library will be used for checking memory leaks.
* Various c++ libraries such as vectors will be used.

## Other Software

* Photoshop will be used to design the graphics.

# Game Overview

Dinosaur Simulator.

## Technical Goals

<3d graphics, 60fps, Challenging AI etc.>

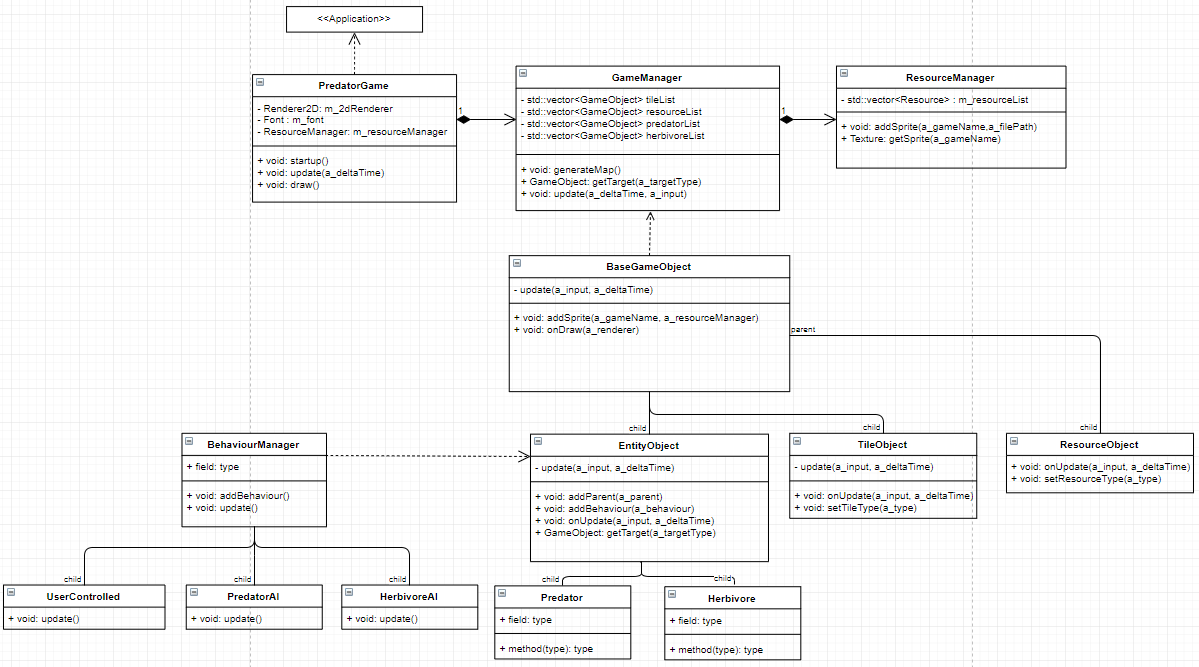
## Game Objects and Logic

<A list of logical elements in the game, i.e. door, button, pistol, ammo, light, bullet, wall, character etc. and description of their behavior and purpose>

## Game Flow

<description of what the player can do (actions) from the start menu to playing the game, through to hitting quit. Include how to win, how to lose, how the player is moved, and what programmer things might need to be considered>

# Mechanics

****

**ResourceManager**

The resource manager stores the sprites in a resource list with a short name that can be accessed by the GameObjects.

**GameManager**

The GameManger generates all the **GameObjects** and their derived classes on the map. The AI and UserControlled Behaviour also access the GameManager through a pointer in order to target specific nodes.

**GameObject**

The GameObject represents the base class of all **agents** and static tiles on the map. It includes a list of base functions used by all of its derived classes.

**BehaviourManager**

The BehaviourManager manages the behaviours associated with the objects. In general, the objects can be AI run (through state machines) or user controlled.

**TileObject**

The Tile represents the non-interactive scenery found on the map.

**ResourceObject**

The ResourceObject represents resources required by the agents such as water, food and footsteps (for tracking).

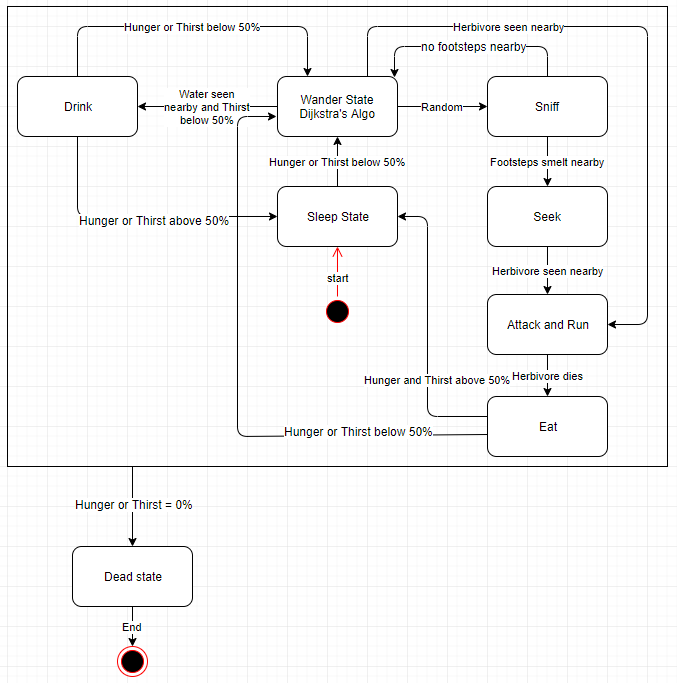
# Graphics

This will be a top-down game.

# Artificial Intelligence

#### **Predator AI**

The predator is represented graphically by a T-rex.

****

**Sleep State**

The predator starts sleep state. It acts asleep and doesn’t do anything.

* If Hunger or Thirst drop below 50% enter the **Wander State**.

**Wander State (Dijkstra’s Algorithm)**

The predator picks a random **node** on the map and begins walking towards it. It uses **Dijkstra’s algorithm** to select the best path to reach the node. I’ve choosen Dijkstra’s algorithm due to the fact there will be less than 40 trees to navigatve around, therefore, the algorithm won’t be too resource intensive. If the predator reaches the node without changing states, it picks a new random node and begins walking towards it.

* If Hunger or Thirst drop below 0% enter the **Dead state**.
* If Water is seen nearby (100 pixels) and water is below 50% enter the **Drink state**.
* If Herbivore seen nearby (100 pixels) enter the **Attack and Run** **state**.
* Randomly (1% of the time): enter the **Sniff state**.

**Sniff State**

The predator performs a sniff which reveals footsteps nearby (100 pixels) on the ground.

* If Hunger or Thirst drop below 0% enter the **Dead state**.
* If there are no footsteps nearby (200 pixels) return to the **Wander State**.
* If there are footsteps nearby (200 pixels) enter the **Seek State**.

**Seek State**

The predator follows each footstep until it reaches the herbivore.

* If Hunger or Thirst drop below 0% enter the **Dead state**.
* If the Herbivore is seen nearby (100 pixels) the predator selects it as a target and then enters the **Attack and Run State**.

**Attack and Run state**

The predator chases the Herbivore constantly attempting to hit it. The Herbivore’s maximum speed can’t match the predator’s maximum speed so the predator will gain ground on it.

* If Hunger or Thirst drop below 0% enter the **Dead state**.
* If herbivore dies enter **Eat state**.

**Eat state**

The predator eats the herbivore then:

* If Hunger or Thirst are below 50% enter the **Wander State**.
* If Hunger or Thirst are above 50% enter the **Sleep State**.

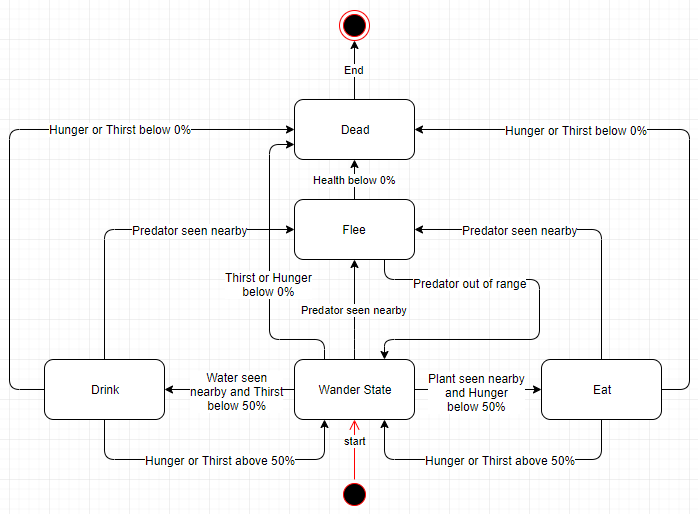
**Drink state**

The predator eats the herbivore then:

* If Hunger or Thirst are below 50% enter the **Wander State**.
* If Hunger or Thirst are above 50% enter the **Sleep State**.

#### **Herbivore AI**

The herbivore is represented graphically by sauropod.



**Wander State**

The herbivore starts in the wander state. It wanders around the world.

* If Hunger or Thirst drop below 0% enter the **Dead state**.
* If thirst drops below 50% and water is seen nearby (100 pixels) enter the **Drink State**.
* If hunger drops below 50% and a plant is seen nearby (100 pixels) enter the **Eat State**.

Drink State

# Physics

The GameManager will manage collision checking of all objects. It will use an Axis Aligned Bounding Box.

# Items Item

There are no item that need to be collected.

# Game Flow

This is a simulator. The GameManager will randomly generate a map and place Predators and Herbivores (prey) on the map. The game will then continue to play randomly generating more herbivores when the number of herbivores drops below three.

# Levels

There are no levels to the game.

# Interface

The interface will be minimalistic. It will consist of a small health, hunger and thirst bar for selected entities.

## Menu

There is no menu.

## Camera

The camera will follow the selected entity.

## Controls

The selected entity is controlled by the mouse and keyboard.

# Asset List

The following graphics need to be created in Photoshop:

* Predator legs, body and head.
* Herbivore legs, body and head.
* Three sets of basic grass tiles to give variety.
* Water tile.
* Three sets of basic trees.
* Footstep graphic.

# Technical Risks

The only technical risk I forsee is running out of time.