Ernie Ledezma

Dani Odicho

Arnold Santos

Cesar Zalzalah

**COMP 565 Project 1**

Our game has a space theme, the black, gray, white colors imply that the person is on the dark side of the moon (Pink Floyd). There are also spaceships floating around from planet Monotron that came to invade the moon. Little did they know, the moon was already flooded with aliens with red eyes that don’t seem very friendly. That’s why the spaceships are just observing the moon and rotating in mysterious way to wait for the right chance. Meanwhile, two humans have made it to the moon in the search for valuable coins. However, when they try to grab the coins something weird happens …>:)

**Additional Models**

Location: GameName1\GameName1Content

**alien.x (Aliens)** - Built with AC3d by Ernie Ledezma

**gold.x (Gold coin)** - Built with AC3d by Ernie Ledezma

**ufo.x (UFO/Spaceship)** –Built with AC3d by Ernie Ledezma

|  |  |  |  |
| --- | --- | --- | --- |
| ***Class*** | ***Type*** | ***Name*** | ***Description*** |
| *Agent.cs* | *variable* | *int treasures* | *# of treasures found* |
| *Agent.cs* | *method* | *Treasures()* | *gets and sets number of treasure* |
| *Treasures.cs* | *variable* | *nav Node* | *Stores an embedded NavNode used to determine distance to treasures* |
| *Treasures.cs* | *variable* | *bool tag* | *Boolean value if treasure has been found* |
| *Treasures.cs* | *method* | *Node()* | *gets or sets node variable* |
| *Treasures.cs* | *method* | *Tag()* | *gets or sets tag variable* |
| *Treasures.cs* | *method* | *Update()* | *moves the treasure to the other side of the terrain when tagged* |
| *Stage.cs* | *variable* | *treasure3D (Model)* | *treasure marker* |
| *Stage.cs* | *variable* | *treasure2 (Model)* | *tagged treasure marker* |
| *Stage.cs* | *variable* | *pathMode (bool)* | *pathfinding/treasure finding mode* |
| *Stage.cs* | *variable* | *TreasureList(List<Treasures>)* | *List of treasure objects* |
| *Stage.cs* | *method* | *LoadScene()* | *Pass TreasureList to player and npagent classes* |
| *Stage.cs* | *method* | *setSurfaceHeight()* | *Modified for interpolation* |
| *Stage.cs* | *method* | *LoadContent()* | *Modified to add treasures and change models* |
| *Stage.cs* | *method* | *Update()* | *add n key listener and added tagged treasures and np mode to inspector mode.* |
| *Player.cs* | *variable* | *int tagDistance* | *tag treasure when user is less than 200 units away from treasure* |
| *Player.cs* | *variable* | *List<Treasures> TreasureList* | *List of treasure objects passed from stage.cs* |
| *Player.cs* | *method* | *Update()* | *move player with arrow keys, tag treasure, move it , and play sound if player gets close to it* |
| *Player.cs* | *method* | *playSound(String path)* | *plays a file located in the bin folder* |
| *NPAgent.cs* | *variable* | *List<Treasures> TreasureList* | *List of treasure objects passed from stage.cs* |
| *NPAgent.cs* | *method* | *ChangePath()* | *switch from following path to following nearest treasure* |
| *NPAgent.cs* | *method* | *nextUntagged()* | *turn towards closest untagged treasure, return false if none exist* |
| *NPAgent.cs* | *method* | *Update()* | *turn toward next untagged treasure, move towards it, tag it, Move treasure, play sound.* |
| *NPAgent.cs* | *method* | *playSound(String path)* | *plays a file located in the bin folder* |

**Added methods and variables in project 2:**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Class*** | ***Type*** | ***Name*** | ***Description*** |
| *NavGraph* | *variable* | *Stage stage* | *Stores instance of the stage* |
| *NavGraph* | *variable* | *Int nodeSpacing* | *Stores the maximum spacing value between the nodes* |
| *NavGraph* | *method* | createQuadTreeNodes(int x1, int y1, int x2, int y2) | *Recursively create the quad tree nodes based on existence of collideble objects* |
| *NavGraph* | *method* | bool objectExists(int x1, int y1, int x2, int y2) | *Checks if there is a collidable object inside the rectangle bounded by x1,y1 and x2,y2* |

**How to run the project:**

1- Extract the AGMGSK.zip file

2- Open the AGMGSK.sln file

3- Click Start to run the project.

**AGMGSK Modified Controls**

The only added control key is ‘k’ which is used to alternate the NPAgent behavior between following the path between the blocks , and walking towards the nearest untagged treasure.

**Treasures Locations**

Locations of the treasures can be found in the stage.cs file:

(400, 400)

(475, 490)

(435, 471)

(500, 425)

The treasures were placed in the relatively flat area near the player to make it easier to test the game. There’s also one treasure behind the walls to help check for the path finding algorithm in the second project

**Terrain Height Algorithm**

A separate project was created to generate the two needed images needed to generate the terrain. The Terrain Map distro was used as a basis to generate the terrain and the createHeightTexture() method was modified. The part that creates the pyramid shape terrain was removed and replaced with an algorithm that generates random height using the brownian motion method. The pseudo code in the lecture notes was used to implement the algorithm which uses a two dimensional array for the height map values. picks random centers then randomly increments or decrements values within the radius. After generating the two dimensional array of heights, the heightToVector4() method was modified to assign colors to heights, the heights were mapped to grayscale colors. Lowest altitudes were given darker color and Higher altitudes were assigned lighter colors. CreateColorTexture() method was also modified to generate three dark colors rather than the grass terrain that it generated initially. After running the project and generating the color texture and height texture images, they were smoothed out by using Gaussian Blur effect in Paint.net. Then the two texture were copied to the AGMGSK project. The

**Implementing Treasures**

We needed to create a new class for treasures that consists of a nav node that holds the position of the treasure and a boolean that determines whether the treasure is tagged or not (along with their getters and setters). Our way of indicating when the treasure is tagged is by moving it all the way to the other side of the terrain. To do that, the class needed to extend MovableModel3d class and override the Update function.

The treasure class is instantiated in the Stage.cs class which creates 4 treasures with specific positions. The class is also used by the Player class and NPAgent class, by calculating the distance of those entities to the treasure. Whenever they are within 200 untis distance of the treasure, the treasure becomes tagged. The treasure runs away, and a sound gets played.

**npAgent Algorithm**

The NPAgent class handles the pathfinding and treasure handling. At first a linked list of treasures is passed in as a parameter in the method, where it gives itself as a value for a private variable inside the class. This allows the list of treasures to be shared between other classes such as stage, player, and NPAgent as well as be modified by any one of those classes.

Next up is the nextUntagged and Update methods. In nextUntagged the list of treasures is traversed, and the distance of each untagged treasure to the agent is measured. When all distances have been measured, the algorithm either turn the agent to face the closest untagged treasure or return a value of “false” if there exists no untagged treasures. To make it more simple, the regular “path” that the NPAgent follows has been simplified to a square and only 4 treasures are placed outside of the square. Different treasures are closer to each regular path node to demonstrate the ability of the algorithm to find the closest treasure.

Lastly we made changed to the update method inside the NPAgent class. The code we added allows for traversing through all untagged treasures, checks the distance between the agent and each treasure, and if any distance is within range, it is marked as tagged in the code and the model just runs away rapidly. When a treasure is tagged, its necessary to turn the agent to face the following untagged treasure in the list. To do this a boolean variable “turn” is set when a treasure is tagged and then the agent turns to face the next untagged treasure.

**Surface Height Algorithm**

The surface height algorithm was implemented in the setSurfaceHeight() method inside Stage.cs class, it was kept there so that it can be used by both Agent.cs and Pack.cs to position the player and the pack properly on the terrain. The method was applied according the the lecture notes of the class. By creating a square of four vector3 nodes A,B,C,D and calculating the heights of those points. then looking at two triangles. ABC, BCD. If the player was located in the upper triangle (ABC) do the lerp according to AB and AC. if the player was in the other triangle. apply the lerp according to DC and DB. The algorithm significantly smoothes the position of the player and the pack of aliens on the terrain.