Hacker Terrain and Treasure Game v0.1

By Ursula Messick and BIlly Morales

Project 1: Comp 565 - Spring 2014

Game Description

Hackers Terrain and Treasure is a cyberpunk game inspired by the movie Hackers (1995). The player (hero) and the npAgent (The Plague) and the run around the cybernet in order to crack the most passwords to get access or block access to the system respectively. Some of the passwords must be found by going into the Gibon. The Gibson is a super-computer represented by the pillars of data that must be traversed in order to find the passwords on the file system. The Digital Obstruction Guards (DOG) are anti-virus applications running around the net that try to block the hero's path.

Github: https://github.com/WebDevGirl/Comp565_TerrainAndTreasure

Models

Name	Modeler	Location
Pillar	Ursula Messick	AC3D **Texture By Gibson Security** http://arstechnica.com/business/2013/12/snapchat-exploit-may-let-hackers-connect-names-and-phone-numbers-in-bulk/
hash	Ursula Messick	AC3D
cracked	Billy Morales	AC3D

Running Application Instructions

The Game

- Unzip MessickMorales565P1.zip
- Open Visual Studio 2014 and run AGMGSK.sln
- Run 'Start'
- Hack the Gibson

Terrain Map

- Unzip
- Open Visual Studio 2014 and run AGMGSK.sln
- Update Stage.cs main() to main2()
- Update TerrainMap.cs main2() to main()
- Run 'Start'

New Key Bindings

Key	Action
n	Runs switchMode() in npAgent to toggle between path finding mode and treasure finding mode

Terrain Generation Algorithm

The terrain generation is done using the Brownian motion algorithm presented in class. The terrain is built around two centers: one at x, z coordinates (70, 70), and another at (400, 400). The algorithm generates a height map by incrementing values around the radii of the centers and randomly walking. The height map is converted to a 1D texture map and then returns a 2D texture map. The algorithm is entirely implemented in the TerrainMap class's createHeightTexture() method.

npAgent Movement

The npAgent runs around the system on a set path known as the terrain path which was provided by the AGMGSK framework.. This in originally initialized and added to the scene at the start of the application. At each update the npAgent moves towards it goal. Once it has reached its goal it goes to the next node in the path. Treasure search mode works in a similar manner. When the npAgent is switched to Treasure Finder mode it loops through a list of treasure objects that is stored on the stage. It looks for the closest non-tagged treasure and then creates a path with that treasure's position via a nav node. The new path, a single nav node path, is generated and set as the path for the npAgent. The original 'nextGoal' is saved before generating a new nextGoal based on the new path.

Once the treasure has been reached (via snap distance) in the Update function the npAgent is switched backed on the terrain path along with its previously saved nextGoal. If the npAgent is switched to Treasure Finder mode and no untagged treasures can be found then the npAgent will stop via the MovableModel3D's reset function. Once they are put back on the terrain path mode the npAgent will run the new MovableModel3D's restart function to set the step size to 1.

Smoothing

The algorithm used for staying on top of the terrain comes from the book XNA 3.0 Game Programming Recipes by Riemer Grootjans. It is implemented in the Terrain class's surfaceHeight(float x, float z) method, which returns the average y value for a given x, z coordinate. The algorithm first finds the relative x and z values to calculate the minimum and maximum height values. It then determines whether the x, z coordinate is in the upper or lower triangle in the terrainMap grid, and calculates the final y value.

Treasures

4 treasures were placed on the map. The new Treasure class extends the Model3D class. Each treasure knows it position, if it was tagged, and who tagged it.

Name	Location (X,Y,Z)
t1	450, 110, 500
t2	446, 110, 450 // wall
t3	348, 110, 330
t4	297, 1550, 340

Classes Modified

Name	Change	Notes
TerrainMap		
	Added createHeightTexture	Brownian motion algorithm
	Updated createColorTexture()	To change colors in terrain
npAgent		
	Imported System.Diagnostics	For quicker Debugging
	Added savedGoal var	To keep track of previous goal when switching modes
	Added mode var	To keep track of current mode of npAgent
	Added terrian_path var	Store terrain path of nav nodes
	Added treasure_path var	Stores single nav node of closest treasure
	Added switchMode()	To switch between Path Finding and Treasure Finding mode.
	Added switchModeToPathFinding()	Function is called in switchMode and a few other places to switch to Path Finding Mode more easily.
	Added chooseClosestTreasure()	Given a list of treasures, find the closest untagged one from the npAgent's current position and return path. If no untagged treasure can be found stop npAgent
	Updated update()	Changed it so that if the goal has been reached and npAgent was in 'treasure finding' mode to switch back to path finding via the switchModeToPathFinding function.
MovableModel3D		
	Added restart()	New function in order to reset model's step size
Stage		
	Added treasures var	To keep track of all the treasures in the scence
	Added Treasures property	So anything with access to the stage (like npAgent) can view the treasure list.
	Updated	- Include the scores of the npAgent and the player and how many

	Update()	treasures they have tagged - Added ‘n’ key binding to switch between Path Finding and Treasure Finding mode
	Added checkForTreasure()	Called in Update(), checks whether an Agent has come into contact with a treasure // If it has, the treasure is tagged // and the tagging Agent's TreasuresTagged is incremented.
	Updated surfaceHeight	To accept float arguments
Agent		
	Added withinRange() call	Checks if an Agent is within range of a Model3D object
	Added treasuresTagged	Field to track treasures tagged
Model3D		
	Added position var Updated addObject()	Keep track of Model3d’s position
	Added updateSprite()	To update images during runtime (like treasures)
Terrain		
	Refactored surfaceHeight()	To use linear interpolation