Created By: Jared Fowler

March 18, 2015

CSUN Comp 565 ; Prof. Barnes

##

ABOUT: Project based off the AGMGSKv6. Notes about added models or modified code can be seen below.

-Theme:

Outer-space, perhaps on the moon. A land composed of mountains, craters, and little green men with top-hats and pipes. ... I'm using dogs for phase 2.

Project 1 ---

-Height Generation Algorithm:

The algorithm used to create the terrain. The algorithm works in a series of steps:
createHeightTexture: 1. Every bit of land is raised up by a specified random range. 2. Via the Brownian algorithm, mountains are formed centred around hard-coded pivots. 3. Craters are then added to the map which, unlike the Brownian, have fixed centers. 4. Noise added to the entire map. 5. The area on which the maze rests is flattened out.
createColorTexture: This function relies heavily upon the function heightToVector4 which uses the height map values to determine the appropriate color texture. Because the terrain is "outer space moon", the terrain is mainly composed of grays and blacks with no particular order.

-NPAgent Movement Algorithm: (Located in file NPAgent.cs in the update function, line 144)

NPAgent is given a reference to the treasure class which keeps track of all treasures on the map. When 'n' is pressed, the Update function will attempt to find the nearest treasure that isn't tagged. If found, a boolean flag is set which indicates that we are now on treasure path mode. Upon every update we check if the treasure is still untagged, or if we are in range (300 px) to grab it. Once a treasure is grabbed or tagged, the NPAgent will simply flip the boolean and resume its original path.

-Staying On Top Of Ground Algorithm: (Located in file Stage.cs, function "setSurfaceHeight" line 345)

The Lerp method was used in order to stay on top of the terrain. This implementation can be found in Stage, function setSurfaceHeight. Data is first gathered to determine the 4 corners of the square and current position. From there, we determine via the distance algorithm which corner the point is closest to. At this point, the Lerp method is applied and a correct height is found.

Project 2 ---

-Packing Algorithm: (Located in file Pack.cs, function "update")

First off, the only modified value which takes effect upon the dog's behavior is the value 'yaw'. The algorithm uses 3 vector2's, each representing one of the forces (cohesion, separation, alignment). Depending on the dog's position, different forces are given to each of these vectors (ie, they are scaled differently.) These vectors are then added together, and the normal is taken. The angle between this vector and the dogs current look out is found, and this angle, multiplied with a small float as to not turn to the desired vector in one update, is output for the yaw value. The algorithm works pretty good, but there still seems to be more tail chasing than I would like.

-Quad-Tree Navigation Graph: (Located in jwfNavigation and instantiated in NPAgent)

Recursive method was used to produce the quad-tree, with a maximum node separation of 3750 and a minimum distance of 150. Some extra functions I added include one which takes care of adding nodes to the dictionary, ensuring that two nodes which might be overlapping do not both get added. A adjacent linker function which ensures that duplicate adjacent nodes are not added.

-A* algorithm: (Located in NPAgent.cs at the bottom of the file. Functionality for showing nodes during path traversing can be found in the NPAgent update function.)

Utilizes a Dictionary for the closed list, a List for the open list, and a stack for the path to traverse. Once NPAgent determines a path, the appropriate nodes are made visible. The open list will be black nodes, open list green, the path white, and the final destination pink.

-Treasure Locations: (Locations refer to original map coord size of 512 x 512)

x	z
393	400
509	493
320	460
447	453
320	493

-Navigation start and end locations (x,z):

Start: (505,490)
End: (495,480)

##

STATUS: March 18, 2015 - Fully functional for phase 2 requirements.

##

USAGE: Refer to the AGMGSKv6 manual for key functions and other details.

-New Functions:

key n:
Causes the npAgent to go after the nearest treasure. Once the treasure is reached, or if the player reaches it first, the Np agent will continue back on its original waypoint path.

key p:
Changes the % of packing. {0%, 33%, 66%, 99%}.

key j:
Causes all the treasure waypoint graph nodes to be drawn.
jey k:
Causes all the treasure waypoint graph nodes to not be drawn.

##

MODELS ADDED:

NAME	AUTHOR	LOCATION
Alien_jwf	Jared Fowler	Local 3dsMax
Tea	Jared Fowler(3dsMax Template)	Local 3dsMax
MoonBaseAlpha	Jared Fowler	Local 3dsMax
Rocket_jwf	Jared Fowler	Local 3dsMax

##

ENUMERATED CHANGES MADE TO AGMGSKv6:

ADDED CLASS FILES:

EyeCandy_MoonBaseAlpha
EyeCandy_Rocket
TreasureList

MODIFIED FILES:

Project 1 --

Stage:
 *   -Changed redraw color to black
 *   -Added my models including moonBaseAlpha and Rocket_jwf
 *   -Changed cloud count to 1, as to act like a mother ship
 *   -Removed Temple from scene
 *   -In function "setSurfaceHeight", utilized the Lerp function
 *   -Treasure list object reference added to variable list.
 *   -npAgent and Player objects changed to include treasure list in constructor
 *   -Added inspector line in function update, to include information concerning treasure count of player and NPAgent

Player:
 *   -Commentary added throughout code for better understanding
 *   -Collidable parameter added to constructor
 *   -If condition added for stage collidable list
 *   -Added a link reference to treasure list
 *   -Added a integer to keep track of treasure count

```
 *      -Treasure count set to 0 in constructor and get/set method added
 *      -Update function has been updated with collision detection with treasures

NPAgent:
 *      -Commentary added throughout code for better understanding
 *      -Reorganized some function calls within update
 *      -Constructor changed to link to a treasure list
 *      -Variables treasreList and treasureGoal added
 *      -Variable treasureCount added along with get/set properties. Initialization in constructor
 *      -Variable treasureListNum added to keep track of which treasure in the list we are after
 *      -Function Update Modified to detect the 'n' key for treasure path. Also updated to handle treasure
path finding and seeking.

Model3D:
 *      -Commentary added throughout code for better understanding
 *      -Integrated and removed second addObject function
 *      -isCollidable now set in constructor with default argument
 *      -Changed fog color to black

Object3D:
 *      -Commentary added throughout code for better understanding
 *      -Line and variable spacing more organized
 *      -Removed second constructor which only differed in scaling capability.
 *       Users will now be forced to input a scaling vector3.
 *      -Constructor now has input parameters for step, stepsize, and bounding
 *       radius.

Terrain:
 *      -Commentary added throughout code for better understanding
 *      -Some type int were changed to UInt32
 *      -Deleted unused constructor (Just to not confuse me with more code)
 *      -Changed fog to black

MovableModel3D:
 *      -Commentary added throughout code for better understanding

Cloud:
 *      -Commentary added throughout code for better understanding
 *      -Some int types changed to UInt32
 *      -Fixed cloud scale in correspondence with making it into a space ship
 *      -Increased height of cloud
 *      -Fixed rotation radians of cloud

Agent:
 *      -Commentary added throughout code for better understanding

Wall:
 *      -Commentary added throughout code for better understanding
 *      -Constructor parameter added "isCollidable" with default value set true
 *      -Converted some int types to UInt32

Path:
 *      -Changed fog color to black

IndexVertexBuffers:
 *      -Commentary added throughout code for better understanding
 *      -Some type int were changed to UInt32

Pack:
 *      -Commentary added throughout code for better understanding
 *      -Some int types changed to UInt32
 *      -Variables initialization taken out of loop for optimization


Project 2 -----------------------------------------------------------------------

NPAgent:
Project: P2 - Navigation and Pack Behaviour
```

```
 *      -Added a check to see if NPAgent is within detection range of a treasure. Located in Update function.
 *      -Added a variable called jwf_path which references a dictionary which contains the nodes used for A*
algorithm path finding. ,
 *       as well as a call to the appropriate constructor. Added it to stage's drawable objects.
 *      -Updated the 'update' function to include a state called "returnPath". This state
 *       occurs after the treasure has been found and NPAgent needs to return to navigation path.
 *      -Added keyboard check for keys 'j' and 'k'. These will enable/disable the visibility of all nodes on
the map by settings flag in
 *       class jwf_path.
 *      -Added A* algorithm function for treasure path finding. Other helper functions are also included at
the bottom of this file.

 Stage:
 Project: P2 - Navigation and Pack Behaviour
 *      -In instantiation of 'Pack', included a pointer to the leader as being he 'Player' object.
 *      -Changed NPAgent argument to make it collidable.
 *      -Added a print statement which prints out the current packing percentage.

 Pack:
  * Project: P2 - Navigation and Pack Behaviour
 *      -Removed unused parameter for constructor. nDogs
 *      -Manually added more dogs and positions of them.
 *      -Added three variables which determine the probability of packing and cohesion factor
 *      -Re-wrote the update function to handle packing. There are mainly 2 calculations that take place,
 *       the first pertaining to Cohesion and Alignment, and the second dealing with Separation.
 *

###################################################################################

UPDATE LOG:
April 2, 2015 – Modified the flocking algorithm… it’s a little bit better now, but there still seems to be
too much tail chasing by the dogs. I feel, however, that the algorithm makes more sense and is overall
simpler now.

March 18, 2015 - Fixed some bugs in the path finding algorithm. Implemented an algorithm to take care of
packing of the dogs, taking into account cohesion, separation, and alignment. Everything seems to function
well, though, I'm not too satisfied with the packing algorithms, but they seem to work fine.  (~4 hours)

March 14-17, 2015 - Successfully implemented and integrated the treasure quad-tree waypoint graph along with
an A* algorithm. These were then integrated into NPAgent's functionality and tested. The nodes are stored
within a Dictionary object, and the A* algorithm uses a Dictionary for the closed list, a List for the open
list, and a stack for the found path. Changes were made to allow for a return path from treasure to
navigation path. The option was added to enable/disable drawing of all graph nodes. In either case, the nodes
which pertain to the A* algorithm will be shown each time a path for NPAgent is calculated. Modified NPAgents
navigation path to go close to each of the treasures. Updated file "Wall.cs" with Prof. Barnes update. (~12
hours)

March 13, 2015 - Read over the requirements for phase 2 of the project. I'm unsure about what is meant by the
quasi-flock method, so I plan to ask about it next lecture. I was able to modify the number of 'dogs' or
aliens running around to be 8 total. They have also been set to have 'player' as their leader. A check was
added in NPAgent which puts it on a treasure path if it is within a certain radius of detection. This was
placed as the first step towards path-finding to treasures via A*. (~2 hours)

February 16, 2015 - After lecture I was able to modify the lerp method. Tested and seems to work good now.
(30 min)

February 12-14, 2015 - Updated the terrain generator in accordance with the minor update provided on the ppt
slides. New terrain was generated.  Wrote individual classes for my models.. These are given the class name
EyeCandy_*, including the rocket and porta-poty. A single cloud now exists which acts as a "mother ship". The
dogs have been replaced with alien models.
                       The lerp method was used in order to fix the y position of moving objects. There are
still some errors which I plan on addressing with Professor Barnes. Modifications were made to the player and
npAgent classes to include treasure path checks, according to the requirements laid out in the document.
Treasure chasing, count, and display has been tested and works. Stage was updated appropriately to
include/reflect these changes.  (~20 hours)

February 10-11, 2015 - A major effort went into studying AGMGSKv6, each file and how it relates to the
others. Also, after becoming more familiar with terrain maker, I decided to go back to a moon type based
```

terrain. Using 3ds max I was able to create various models, and by installing a plugin for 3ds I was able to directly export direct x files. It was a bit of a pain to get it to work at first. Direct X runtime and Visual Studio 2008 were installed first in order for the added plugin to work. As part of going over AGMGSKv6, I added a bit of commentary and rearranged some of the code. I still haven't really written any of my own code, but I feel more familiar with the framework and will start to write code next time. (~15 hours)

February 4-5, 2015 - A great deal of study went into understanding the terrain maker better. I decided against a space theme and shifted more towards a highland theme. (I'll probably still set the sky to be a space like view.) The Brownian Algorithm was implemented along with a few structures which make the functions extremely more modular. The algorithm makes use of both diamond and circular shapes. Both a height and color map were produced, and blurred via Gimp using the Gaussian Blur. (~5 hours)

February 3, 2015 - Downloaded AGMGSKv6, along with the texturemap creator. Spent a good while looking over the texturemap code. I feel more comfortable with how it works, and will probably attempt to create my own terrain next time. My idea is to have a space-like environment. In order to get the terrainmap program to work I had to add a reference for System.Drawing. (3 hours)

January 28, 2015 - Examined the program provided by Professor Barnes called "axis". Points of interest include the following: 1. As opposed to openGL, XNA's matrices are multiplied in reverse order. This is because XNA's matrices are row-majored versus openGL's which are column major. (2 hours)