P1 due 3/8/2015 Update # 1 Model, terrain and treasures 565 Barnes Sp 2015

This assignment (and the second assignment) uses the AGMGSKv6 distribution. The AGMGSKv6 distro has 3 projects: MatrixInfo, TerrainMap, and AGMGSK. The source files, content, and contentXNB of each project are in their respective subdirectories. There is an AGMGSKv6Doc.pdf documentation file in the distribution that describes the starter kit. Your assignment should be done inside a project named AGMGSK, or you will need to change the namespace for all the *.cs files (refactor – your choice) in the AGMGSK project distribution.

Scene design

You should first decide what the theme of your scene will be. All scenes must have a rolling hills (no really sharp inclines – all terrain is walkable by agents). For example you could have a scene with Stonehenge like ring structures, a desert with pyramids, the American plains with teepees, or a city scene with blocks of simple rectangular building. You can populate your scene with models that you load. These models should be generated with AC3D or a modeler of your choice that can save (export) direct x files (*.x). The direct x files should be triangulated and saved with material and normals. My advice is to keep it simple for the first project and avoid textures in your models. If you use AC3D's File | export | Direct X, in the export dialog select "right handed coordinate system". Some of your models can be downloaded from the web (they must be scaled appropriately) I want you to get a little experience with modelers. You do not need a lot of models. Do not spend too much time on models and scene design – this is just eye candy.

AGMGSK is scaled so that 4 pixels = 1 inch. The spacing between vertices in your terrain will be 150 pixels. Thus your terrain will range from 0 to76,800 in the X and Z dimensions. The origin of the scene will be the left, back, corner of your terrain when viewed from above (+Y). For each step (1) and Agent takes the step size is 10.

In your scene you should have 4 modeled treasures. The treasures should be scaled in size between 100 by 300 pixels in width, height, and depth. One of the treasures must be placed inside the walls close to the inside corner (vertex (447, 453) position (67050, 67950)) but not within any wall “brick” bounding sphere. Treasure objects are the goal of the "game". Both the "chaser" (a Player) and the "evader" (an NPAgent) can “get” treasures. The "chaser" moves under Player's input controls. "Evader" moves by its own algorithm (NPAgent's).

Terrain generation

Once you have a theme you need to generate your terrain by modifying the TerrainMap.cs program (TerrainMap distro) and then using a paint program's Gaussian blur effect to "average" or "smooth" the height and color textures.

1. Using TerrainMap.cs program. You should use the Brownian-Motion terrain generation algorithm presented in lecture to create height values. You should have some nearly flat terrain the lower, "testing", quadrant of the scene (X and Z > 38,250). The area of the "walls" should be flat ("relatively flat"). Think about how your step and radius parameters affect the dispersion of height values.

Color Table. You should design a color table that will map height values into colors. Since we are using textures to hold the height values that range from 0 to 255. You could have a different color for every

interval of 25 or 50 heights. For example, height values of 0 could be a "tan or sand like color", and 1 to 25 could be a tan-green, or perhaps a yellow-green, 26 – 50 could be a darker green, above 225 you might have white for snow. You should add some noise to your vertex color values.

Smoothing. You should smooth your heightTexture and colorTexture. You can do this with a paint tool like paint.NET or Gimp to add Gaussian blur effect to your textures. This will make height and color transitions smoother.

Put the heightTexture.png and colorTexture.png files in the Content directory of your P1 application (AGMGSK project) after smoothing. For a MonoGames projects that use contentXNB assets put these files (heightTexture.xnb, colorTexture.xnb) in the debug Content directory of your MonoGames project. You can convert the *.png files to *.xnb in JD 1618 with a simple XNA application.

Navigation -- terrain following. Once you have your terrain generated and your models placed. You need to modify the starter kit so that the player object and NPAgent object move better on top of the terrain. In the distribution Agents and Pack object3D's are set at the surface height of the minimum (X, Z) vertex for the surface they are on ("upper left corner of quad holding two surfaces"). Terrain following can be done by interpolating with the Vector3.Lerp(....) method. Optionally you can have an 'L' key toggle Lerp or default terrain following on/off. This way you can see/test the effect of Lerp.

NPAgent. The NPAgent's update method should be modified so that it moves in one of two states: "path-following", or "treasure-goal". Path-following is the default state and is implemented in the AGMGSK distro. In the treasure-goal state, the NPAgent moves towards the next closest unfound treasure until it "tags" the treasure. When the user presses the 'n' keyboard key the NPAgent state should change from path-following to treasure-goal state. The NPAgent should remember what its current path-following goal is, so it can resume path-following. The NPAgent in treasure-goal movement should always go to the closest untagged treasure. When the NPAgent "tags" a treasure it automatically switches back into path-following mode and resumes its interrupted path. It finds 1 treasure (if one is not tagged) for each 'n' press.

Either the Agent or NPAgent can "find" or "tag" a treasure if it gets within 300 pixels of a treasure. Once a treasure has been found it should be "tagged", so the treasure is no longer active. Its display should indicate its "tagged" (non-active) state. The Agent that tagged the treasure increases its treasure count. Your program should display the number of treasures found ("tagged") by each agent in an Inspector pane info pane. Consider placing the treasures in the flat "testing" area where the Player is loaded. This way you can see and test your program quickly without having to wait for the NPAgent to move relatively long distances. The simulation does not end when all treasures are tagged. The program ends when the user closes the window or presses the 'esc' key.

Do not change the assigned key bindings in AGMGSK.v6. I don't care if your preference is for "awsd" over arrow keys.

What to submit.

Submit a zipped archive of your project's directory(ies) on class's Moodle site. The archived submission file should be named <LastName1...LastNameN>565P1.zip for zipped file. The "LastNames" should be listed alphabetically for the group. For example, "BarnesSmith565P1.zip". You should submit entire project subdirectories. You need to compress (*.zip) all your submission files in your project. You can submit 2 compressed files: (1) for the terrain mapping and (2) for the project. Submit complete zipped projects without version control.

The project name, class, and the names and email of all group members must be stated at the beginning of Program.cs and all AGMGSKv6 *.cs files you modify in a comment header (not models). Add your comment in its own comment block above the copyright notice in AGMGSKv6 files. Internal source file documentation (comments) should be done for a knowledgeable programmer and environment user. "What would you want to read 6 months later to remember what you did and why." **Do use installers for your project**. I do not want to install any student software on lab or personal computers. You are responsible for your software being able to run on systems equivalent to the ones in the classroom. Running on your home system or laptop **IS NOT** sufficient testing. If you are submitting a Mac OSX project be sure to talk with me about possible modifications, so I can run / grade it.

Please submit electronic documentation as part of your compressed submitted file -- *.doc, *.docx, *.pdf, *.rtf. The documentation must have the following items:

A description of the theme of your scene.

A list (or table) of the models you added to the scene and their source (author, location). Models you make would have your name and the modeler used for the location. Models downloaded would have their author and URL.

The documentation should also describe how to run the program -- did you add any user input options? You must provide a mapping of all new key mappings for user input. Again, do not change the existing and required keyboard commands. Please provide the location of the 4 treasures (x, z) in your documentation.

You must provide a description of your height generation algorithm, your NPAgent's movement algorithm, and your algorithm for staying on top of the terrain. In each description be sure to name the methods (method signature, containing class) modified to implement your solution. **Please include a table of AGMGSK classes modified (what variables added, what methods added or modified).**

The harder (searching for relevant methods, finding and installing obscure archivers, having to correct your project) it is to evaluate / grade your project the lower the grade. I will probably just return a grade of "upgradable, correct and resubmit with penalty".

No late submissions are accepted unless I allow an extension. Examples of valid late submission reasons are: medical problems, accidents, work required travel. From my perspective requirements from other classes have a lower priority than assignments from this class. I am reasonable about extensions. It is your responsibility to provide supporting evidence for your reason and to request an extension before the due date. You have several weeks to manage this project, you can notify me of problems before the day its due. Incomplete projects should be submitted on the due date and will be graded.

Partial Submissions. If you submit an incomplete project you must still provide documentation that describes what is wrong or missing in the submission. What steps need to be done next, and briefly how you would do these steps. It is assumed you document your work as you do it: design, implementation, and verification.

Team Work. You can work in teams (1 to 3 members) on projects. The goal of team work is to share the work. I hope that you will discuss design and implementation issues. Help each other with problems and learn from each other. If you work in a team it is not acceptable for one team member to do all the work -- you must work together and understand the project. There is one project submission per team and all team members have the same grade. The project's problems will be used as examples for questions on exams.
If a team can no longer function together they should "divorce". In a divorce each member shares the existing design and implementation at the time of the divorce and works separately to complete the project. Divorces should be described in the project's initial comment block and in the "readme" documentation. Divorces do not join another group, they become a group of one. Work well and fairly with each other.

Reusing code

I encourage you to discuss problems you have on your projects with others (teams). Look at each other's code and displays. Communicate and help each other. Try and learn the name of, and talk with, every student in the class!

Your programs should be your (team's) work. **You should not copy from other groups.** Solutions may be similar - but should not be identical. While programs should not be copied, some code can be reused -- from many sources. All reused code must be referenced and used with permission from the author. A brief reference to the author should be done in a comment preceding the usage in the listing. There should be a complete reference in the initial program's comment block containing the author's name and the source -- like you would reference a source in a term paper. Reused code cannot be for major parts, or concepts in the assignment. For example, you can't use someone else's collision detection code, or someone's physics simulation library. Failure to provide reference to published, reused code will be considered copying and negatively affect the project grade. All files in the ABMGSKv6should keep their copyright notice. You don’t have to reference any use of code distributed in AGMGSKv6 – that is assumed.