Assignment $^{\#}$3: Game Project

Due Date: See *Deliverables*, below

The remainder of the semester will be spent developing your own video game. Guidelines and requirements for the game implementation are given below. The end result will be a completed game which you will demonstrate at the end of semester "game party".

The project is intended to be done in groups of two. If you prefer to work alone, you may. But the assignment is designed assuming it would benefit from two students working together on it. The requirements and quality expectations are the same if you work alone or with a partner.

Project Requirements

For full credit, your game must include each of the following features. A few of these are topics we have already covered, but many of them are things we will be going over in the coming weeks.

- **External models**. The game must include at least two custom-made models designed and created by you, imported as OBJ files. *Each person in the group must contribute at least one of those external models.* Blender or Maya are recommended. You will need to familiarize yourself and possibly adjust for the limitations of the TAGE OBJ importer. If you modify TAGE's OBJ importer (or any other TAGE code), make sure that it is only with *your own* code.

  (related) – Each of these models must utilize a texture of your own design using UV-unwrapping. *The texture must be related to the model shape, and not just be a random or repeating pattern.*

- **Networked Multi-player**. Your game must be playable by multiple players (at least two) over the web in the lab. Players must be able to see each other's avatars when nearby and facing each other. It is *also* a requirement that your game function in *single-player mode* (see below).

  (related) - Players must have a way of selecting their character avatar at startup, and the selection must be conveyed to other game clients so their ghosts use the proper avatar for each player. This means if you use the simple protocol discussed in class, it must be extended to pass a model or texture name as part of the CREATE message sent to remote clients.

- **Skybox and Terrain**. Your game must include a *skybox* and *terrain*. You should use the TAGE Skybox and Terrain classes, or modified versions. Note that it is <u>not</u> a requirement that these components be integral to the gameplay. For example, if your game takes place mostly "indoors" then you can meet this requirement by providing a way for a player to "go outside" and move around on (and up and down on) the terrain, bounded by a skybox, which has no direct effect on the game – but the program must at least demonstrate use of a skybox and a terrain.

- **Lights**. You must use the TAGE lighting classes in your game. In particular, you must make *effective* use of <u>at least two</u> lights (positional and/or spotlight) in addition to the global ambient light. At least one must be possible for the player to turn on/off. You are also encouraged to add additional types of lights to the TAGE engine (such as directional), but this is not a requirement.

- **HUD**. Include a HUD component to let the player know what is happening in the game. If your game is better without a HUD, you may include an input to enable/disable the HUD display.

- **3D Sound**. Your game must include several 3D action-specific game sounds as well as background sound. Use TAGE's audio package (which uses JOAL), or use JOAL directly. The more action-specific sounds the better, but you must have at least two.

- **Hierarchical SceneGraph**. There must be a set of objects that makes use of the scenegraph to build a hierarchical object or system. You must use TAGE's scenegraph support for this. Be sure to select the appropriate sets of matrix propagation for your situation.
- **Animation**. Add skeletal animation using keyframes for at least one model in your game. It does not have to be complex, but should look correct. The animation must be created by you, using Blender. You can then use the RAGE Blender export tool and import it into your TAGE game.
- **NPCs**. Your game must include one or more "Non-Player Characters" managed by an AI controller. This means the NPCs must do something under AI control, not just stand around frozen, and not behave completely randomly. We will study various approaches in class.
- **Physics**. Your game must use the TAGE *physics engine.* You may use TAGE's built-in physics package (it uses JBullet), or you can use JBullet directly. You should also make use of JBullet's ability to detect which objects have collided, as shown in the distributed example. You may use the physics engine for all of your objects, or for just a subset of your objects. You may use TAGE's physics *visualization tool* during development, but it should be disabled for game play (if you wish to retain a key that enables/disables physics visualization, that is ok).

Additional Notes

- Your game may be based on first-person (1P) or third-person (3P) view. 3P is preferable.
- Your game must support keyboard and gamepad input.
- Your game must function in both FSEM (full screen exclusive mode) and windowed mode.
- You may use any network architecture you like for your game. A client-server "fat-client" approach with UDP, using the TAGE networking package, is likely to be the easiest.
- You are encouraged to incorporate additional models or assets (beyond those that you are required to create yourself), taken from any (legal) source. As in the previous assignments, your accompanying document must include evidence that you are allowed to use the assets.
- It is a requirement that your game have a "single-player mode", even if it is designed as a fully multi-player game. Specifically, it should be possible for a single player to be able to run around in the world and see its features (at least the ones that don't rely on networking), without connecting to the server. Note that it is *not* a requirement that the game be very logical in this mode; for example, it's not necessary that there be a way to "win" in single-player mode. And, the game would not be expected to display *ghost* avatars when in "single-player mode".
- All resources required by your game (e.g. texture files, sound files, etc.) must be accessed using *relative-paths* (that is, don't hard code absolute pathnames into your program). In most cases, this will mean using the standard TAGE asset folders.
- Specify the server's IP address in the client run.bat file, do <u>not</u> hardcode it in your Java code.
- Your program must run correctly on at least TWO of the machines in RVR 5029 (it's networked), and your submission should specify which two machines you tested your program on.
- You may change the name of the top-level package to something matching your game's name (rather than the generic "**`a3.MyGame`**"). *Document the package name in your submission*.

When satisfying the above requirements, don't just copy the examples from class. In each case, try to find ways of using the techniques in a manner that fits *your* game and its theme.

Deliverables

There are several deliverables for the Project, including progress *Milestones*, each with its own due date and requirements. Each team arranges to meet with the instructor (or grader) to demonstrate completion of Milestones during the indicated weeks (we also might use one of the Zoom class sessions for this). Scores will be assigned based on satisfactory progress (meaning solidly working, but possibly not in its final form) of each Milestone separately.

Milestone 1: Networking, Skybox, Terrain, UV-unwrapped Models. Week of April 8 – 12.

Milestone 2: Physics, Sound, NPCs/AI & Animation. Week of April 29 – May 3.

Game Party: Be prepared to give a demonstration of your working game at our GAME PARTY (during our scheduled Final Exam period). Each team will have about 5 minutes to show the game in action. If possible, we will also try to facilitate playing each others' games.

Final Code and Report. DUE: Saturday, May 11th at Midnight (there is NO 24-hour late period). BOTH team members are to submit to Canvas the same identical single ZIP file containing the following set of deliverables:

- Both the source code (.java) and compiled (.class) files for your game, and for TAGE. Updated javadocs for TAGE should also be provided.
- Batch (.bat) files for compiling and running your game. You may provide separate batch files for starting the server, and for starting clients. You also will need a .bat file to compile TAGE.
- A folder containing all the resources required by the game (texture maps, sound files, model files, etc.), *organized in the zip file in the way they need to be accessed by the game*. (this can be satisfied by utilizing the "assets" folder as we did in assignments #1 and #2)
- A PDF guide (~3-5 pages) to your game with the following 16 numbered items, in this order:
  1. the name of your game, your names, and your 165 section number(s)
  2. at least one image (screenshot) showing a typical scene from your game being played
  3. instructions for compiling and running your game, including the network server
  4. any special device requirements, such as particular input device(s)
  5. how to play your game, including what things happen and how the scoring works
  6. what player controls are available (what all keyboard/gamepad buttons do, etc.)
  7. a brief summary of any changes (or none) that you made to the network protocol
  8. a list of changes and additions that you made to TAGE
  9. a statement indicating the (1) *genre*, (2) *theme*, (3) *dimensionality*, and (4) *activities* utilized in your game (see week 1 notes -- chapter 00 -- for examples)
  10. an explanation of where (in the game, not the code) each project requirement is satisfied
  11. a list of the requirements that you *weren't* able to get working
  12. any technique you used in your game that goes beyond the requirements
  13. the contributions of each team member, including who designed which model(s)
  14. a list of assets that you created yourself (models, textures, heightmap, etc.), and items obtained that were distributed in CSc-155 or CSc-165.
  15. Source and evidence of permission for any item (models, textures, etc.) not listed in #14
  16. which RVR-5029 lab machine**s** (at least two – it's networked!) on which your program was tested and is known to work correctly on.

All submitted work, other than the use of public-domain models, textures, sounds, etc., as described above, must be strictly your own or that of your team partner.

APPENDIX A

Previous requirements, and how they apply in this project:

- Constructing a manual object from scratch. Not required, but allowed.
- Instantiating built-in TAGE objects/assets (such as the dolphin). Not required, but allowed.
- Input actions handled by Action classes. Still required.
- Keyboard and gamepad controls. Still required, but can be substituted with other controls.
- Controls based on elapsed time rather than fixed amounts. Still required.
- Display world axes. Not required, but useful for testing – recommended as an option, as well as a capability for disabling them.
- Detecting and handling collisions. Handle via the physics engine.
- Package organization. Still required except that the starter package can be given a better name.
- Ability to run via batch file. Still required.
- Camera controller. Still required.
- Multiple viewports. Not required, but useful to make many games more interesting!
- Ground Plane. Not required for game play, but there must be terrain (and terrain-following) somewhere in your world.
- Scene node controllers. Not required, but recommended.
- Hierarchical scene node(s). Still required – must include some hierarchical structure.
- Avatar. Still required – additionally, ghost avatar(s) and NPCs are also required.
- Player's Guide. Still required as part of final report.

APPENDIX B

Grading breakdown:

Milestone 1	7
Milestone 2	7
Game Day demo	6
Final Submission	30
total:	50 points