Assignment #1: Building a Game and Extending a Game Engine

"Dolphin Tour" ----- Due: Thursday Feb 15th (3 weeks)

The assignment requires you to use – and modify – your own copy of the TAGE game engine distributed in class, and use it to build a simple game.

The game you will make is called *"Dolphin Tour"*. The player uses the keyboard and gamepad (Xbox controller or equivalent) to fly around in outer space on the back of a dolphin, visiting tourist sites. The sites are four objects scattered randomly around the space, and to visit one, a player has to ride the dolphin up very close to the site and run into it. This causes a prize (a refrigerator magnet) to be collected (and placed somewhere), and the score to be incremented. The game is over when all sites have been visited (i.e., when the score is 4). The score and elapsed time are shown on a HUD.

The camera generally stays on the back of the dolphin, but it is allowed to "hop off" and back on.

Game Program Requirements

- Your game extends **`VariableFrameRateGame`**, and overrides at least **`loadShapes()`**, **`loadTextures()`**, **`buildObjects()`**, **`initializeLights()`**, **`initializeGame()`**, and **`update()`**. The game's logic is to reside in **`update()`**, or in functions invoked by **`update()`**.
- Your scene should instantiate four "sites" (game objects), randomly positioned in the 3D space. You should use the built-in shapes *Sphere*, *Cube*, *Torus*, and *Plane* for the sites, and texture them. You should also use **`scale()`** to set the objects to various sizes. Two of the textures must be made by you, and two must use textures obtained from some other source.
- Your game must include classes that extend **AbstractInputAction**, and then utilize **`InputManager.associateAction()`** to link device components with your **Action** classes.
- At least the following key mappings must be provided (you can add more if you like):
  - **W / S**: move *forward / backward*. They *move* the dolphin along of its local forward axis.
  - **A / D**: turn *left / right* (yaw). These keys *turn* the dolphin left/right around the world Y axis.
  - **Up-arrow / Down-arrow**: *turn* the dolphin around its own side axis ("pitch").
  - **SpaceBar**: toggle between being ON the dolphin, and OFF the dolphin. When using this key to hop OFF the dolphin, it should move the camera axes and position near the dolphin. There isnt a game-related reason to hop off the dolphin, it is added here as a requirement to help you familiarize yourself with the TAGE camera, which will will use more in the next assignment.
- At least the following controller mappings must be provided (you can add more if you like):
  - **Y-axis**: move dolphin forward and backward in the same manner as the W and S keys
  - **X-axis**: yaw (turn) the dolphin left and right in the same manner as the A and D keys
  - **YRot-axis**: pitch dolphin around its side axis, the same as the Up/Down arrow keys (if using a Logitech gamepad, it will need to be in "X" mode for YRot axis to be available).
- New *yaw()* and *pitch()* functions should be created and added to the **GameObject** ~~and **Camera**~~ class, and used for the inputs described above. Yaw should be implemented as *global yaw* (relative to world axes), and pitch is to be implemented as *local pitch* (relative to object axes).

- The game must display X, Y, and Z **world axes** showing where the world origin is located. You should build these axes as GameObjects using the Line shape in TAGE.
- The **HUD** score should change to "You Win!" when all four sites have been visited.
- You must keep the dolphin from going too far from the camera when the player is "off" the dolphin, by not allowing a dolphin move if the resulting distance would be more than some threshold.
- Your game should include ambient light, and at least one positional light.
- You are to add t``wo additional features of your own choice:
  - an additional game activity not listed above. For example, the dolphin could be required to periodically eat some food. The added activity is *your choice*, so long as it isn't too trivial.
  - an additional game object, hand-built using TAGE's **ManualObject** class. For example, you could design a bin where the refrigerator magnets are collected. Or, you could add a monster that attacks you when you are off of the dolphin. These are just ideas, try making up your own!
- You will need to design a "refrigerator magnet" object that is generated when a site is visited. You can use whatever shape you want (a cube or plane would be logical), and textured appropriately. The magnet then needs to be placed either on the back of the dolphin, or in a bin that you design.

Additional Notes

- You'll need to keep track of whether the camera is ON or OFF of the dolphin. When getting OFF the dolphin, your program should move the camera position to an appropriate location near the dolphin, and facing the same direction as the dolphin.
- Sites are "visited" by running into them with the dolphin. This means you must do some simple collision detection between the dolphin location and the sites. For this assignment, you can simply check each site object to see if its distance to the dolphin is sufficiently small. Later in the semester we will do more sophisticated collision detection.
- If you don't have a controller that implements X, Y, and RY axes, check with your instructor – he has a few extras available for checkout. Cheap ones can be purchased for about $25 so you might consider getting one for yourself if you don't already have one.
- Your dolphin movements must be computed based on elapsed time, so that their speed is the same regardless of which machine is being used.
- If you want to have other objects in your world move around, you can use the built-in TAGE rotation controller, or you can try retrieving an object's position and rotation vectors, and changing them. If you try the latter, you should notice that each node has "local" and "world" transforms. You can use the "world" transforms to look up the position and orientation of an object. You can change an object's position or orientation by altering the "local" transforms, but do not modify the "world" transforms (we'll learn more about "local" versus "world" transforms later).
- VERY IMPORTANT → Make sure that your program works on at least one workstation in the RVR-5029 lab. Don't wait until the last minute to test this!!! The same goes for being able to run your program from the command line. I have no easy way of grading your homework if I can't easily launch it from at least one of the lab machines!

Deliverables and Coding Style Requirements

- This is an INDIVIDUAL assignment. You may use any of the code distributed in class, but you may not use code obtained from the web, fellow students, or elsewhere.
- Submit to Canvas TWO items:
  - (1) a ZIP folder with your A1 game submission and game engine, as described below
  - (2) a TEXT file (.txt) indicating which RVR-5029 lab machine you tested your program on
- The ZIP folder must be named with your full name, as *lastname_firstname*. Inside this folder are:
  - ✓ (1) a subfolder "**tage**" that is the package containing your version of TAGE
  - ✓ (2) a subfolder "**a1**" that is the package containing your game application
  - ✓ (3) the "**assets**" subfolder
  - ✓ (4) batch files "**compile.bat**" and "**run.bat**"
  - ✓ (5) a PDF **readme** document.
- Your game code is to be placed in the "**a1**" folder, which is the Java package for your game (do not call this package folder "myGame" as was done in the sample program). Your main program that extends VariableFrameRateGame should be in a file inside of a1 named **myGame.java**.
- The "**tage**" folder contains the complete TAGE game engine, including your changes. Your new *pitch()* and *yaw()* functions should, for example, be found in the GameObject and Camera classes. (It is NOT necessary for the "tage" folder to include any of the `.jar` files that TAGE relies on.)
- You are encouraged to make other changes and additions to your copy of TAGE – however you should document any such changes in your **readme** PDF file.
- Your `compile.bat` file should successfully compile BOTH your TAGE engine, and your a1 game application.
- Your `run.bat` file should successfully run your game, using the command given in HelloDolphin:

```
java --add-exports java.base/java.lang=ALL-UNNAMED --add-exports java.desktop/sun.awt=ALL-UNNAMED --add-exports
  java.desktop/sun.java2d=ALL-UNNAMED -Dsun.java2d.d3d=false -Dsun.java2d.uiScale=1 a1.MyGame
```

- Your "**readme.pdf**" document (approx. 2 pages) must include the following 9 numbered items:

1. your full name and CSc-165 section number, and "A1 – Dolphin Tour"
2. a screenshot (JPG file) showing a typical scene from your game
3. how your game is played, a list of the inputs and what they do
4. a description of your additional "game activity"
5. a description of your additional "game object"
6. a description of your "refrigerator magnet" and where they are placed
7. a clear list of all changes you made to the TAGE engine
8. a list of any requirements that you *weren't* able to get working
9. a list of anything special that you added beyond what was specified in the requirements
10. a list of every asset used in your game, and whether you made it. For each asset that you didn't create yourself, indicate where you got it, and provide clear evidence that it is legal for you to use in this game (such as written permission, or a posted license). If an asset was copied from the distributed TAGE examples, just state that.

Note that the submitted files must be organized in the proper hierarchy *in the ZIP file*, as indicated in the coding style requirements listed above, *except* the TEXT file which must be *outside* the ZIP folder.