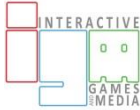
	<p style="text-align: center;"> Rochester Institute of Technology Golisano College of Computing and Information Sciences School of Interactive Games and Media 2145 Golisano Hall – (585) 475-7680 </p>	
---	--	---

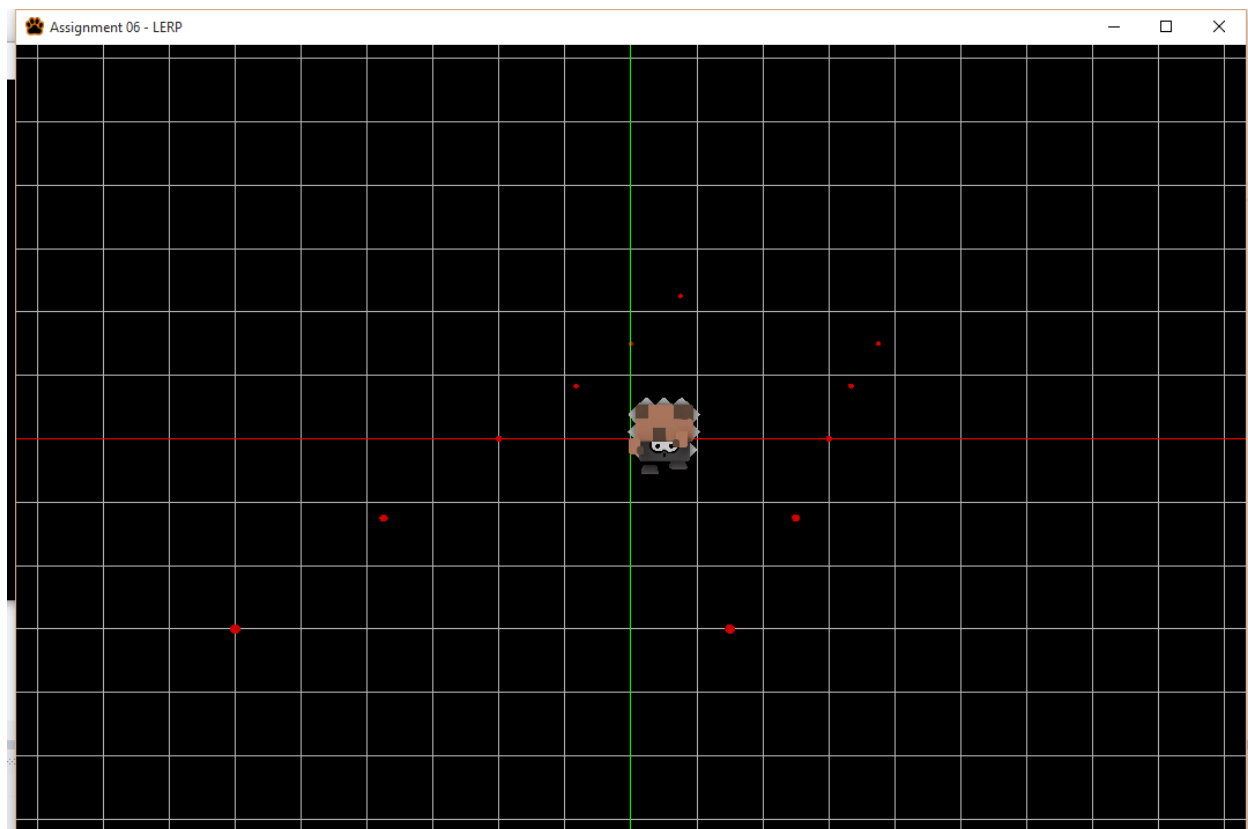
Data Structures & Algorithms for Games & Simulation II
IGME 309, 2015 Fall
A06: LERP

Due: September 27th 2015 at 23:59Hrs.

Instructions:

You have starting code through ReEngine, you may use your own solution if you want to, but the translation of starting code is entirely your responsibility. Startup code is available at: <https://github.com/labigm/ReEngApplication> under the **A06_LERP** project. An example binary may be found under the _Binary folder as usual.

The goal of this homework assignment is to get practice performing linear interpolation on an object through different segments.



To keep things interesting an animated model is already loaded in the scene for you in the startup code but you don't need to do anything to it other than modify the transformation matrix that represents its position in the world in the update method though the mesh manager like so:

```
m_pMeshMgr->SetModelMatrix(<InsertYourMatrixhere>, "WallEye");
```

The list of locations that define the movement of the character is the following:

```
vector3(-4.0f, -2.0f, 5.0f);  
vector3(1.0f, -2.0f, 5.0f);  
vector3(-3.0f, -1.0f, 3.0f);  
vector3(2.0f, -1.0f, 3.0f);  
vector3(-2.0f, 0.0f, 0.0f);  
vector3(3.0f, 0.0f, 0.0f);  
vector3(-1.0f, 1.0f, -3.0f);  
vector3(4.0f, 1.0f, -3.0f);  
vector3(0.0f, 2.0f, -5.0f);  
vector3(5.0f, 2.0f, -5.0f);  
vector3(1.0f, 3.0f, -5.0f);
```

You may use any data structure to store these locations but points will be subtracted if they are not grouped under a single data structure (i.e. you just create n different variables).

To keep things simple all translations between locations will take the same amount of time no matter what is the distance between them. By doing it like this at some routes the translation will go faster THIS IS TOTALLY OK. Extra points will be given if you make the translation uniform though the whole route while using time and not processor cycles.

To make visualization easier you need to add spheres of 0.1f in size wherever your locations are.

Release() does not need any changes unless you allocate new pointers.

Provided to you:

- **float** fTimeSpan - A variable that stores the difference in seconds between render calls.
- **float** fRunTime - A variable that stores the amount of time the program has run.
- **float** fDuration - A variable that stores the time a transition between two locations should take.

Hints:

- My solution was perform in under 50 lines of code but it can probably be improved upon.
- You can use the glm::lerp function and the MapValue function
- You do not need to worry about the character's animation WHATSOEVER

Requirements:

- Your code MUST compile AND execute. I will not take points out of the program if it doesn't compile AND/OR run, I will simply not grade it. If your program does not run it will receive a 0/100. If you are having trouble with something in the code comment out the lines, say what you wanted to do and what you suspect the issue is. That will result in partial credit, which is better than not having a grade.
- Memory Leaks are acceptable, points will be taken off, but the code will be reviewed. NO MEMORY ALLOCATION WAS REQUIRED FOR MY SOLUTION
- You get rid of the "trash files" (intermediary files). (Z_Delete folder AND sdf file)
- Zip your solution and upload it to the dropbox in my courses.

Grading:

(-???) Cheating

(-100) Code not compiling or executing.

(-10 to -20) Memory leaks (You are not reserving new memory for this test so this shouldn't be an issue)

(-30) For each uncommented method. I need to know what you are doing or trying to do.

(-10) You forgot to delete the Z_Delete folder

(-10) you forgot to delete the .sdf file

(-20) Your solution doesn't loop through the points after they are done (the sequence goes from 0 to n to 0 to n to 0 to.... Etc.)

(-50) Your solution is not time-based.

(-20) Your enemy skips points while moving.

(-80) You hardcode the movements somehow.

General solution tips:

Think about it, you already have the list of point on which the character will be, you can take point 0 and 1 and calculate the position based on how much progress the timer has done, progress is a measurement from 0 to 100% (or 0 to 1) but it's actually coming from 0 to fDuration; once fRunTime is larger than said duration it means that is done with the first sets of points; and this is true for every set of points; the last case going from the last point to the very first one could be the most tricky case.

Extra points:

+15 Your movements are at a constant speed though the whole list of points without being processor dependent.