|  |
| --- |
| INFO6044 Game Engine Frameworks & Patterns  Project #2 Fall 2023; instructor: Michael Feeney |
| Weight: Project weight evenly distributed across the “project” portion of marks |
| Due Date: Friday, December 15th at 11:59 PM |

*Note: This project can be done alone or in pairs*

Description and Purpose

You are to create an OpenGL application that displays a “reasonably complex” dynamic 3D\* scene being controlled by an externally loaded “animation script”, working completely in Lua, or using Lua as a key scripting component, to exercise functions of an animation system of your own design.

Alternatively, you can use another “C/C++ bound” language like python, or your own custom scripting language.

The main focus of the project is to reliably handle motion and time when objects are moving with non-constant velocity, and be intuitive and straight-forward enough for non-programmer using the script. More details below.

( \*This can be a “2D” scene – i.e. limiting your view, etc. to 2D dimensions, but *is actually being handled in a 3D environment, and using 3D models* - but *your script must have the ability to handle all three dimensions*. The exception to this would be things like orientations, etc., that are significantly more difficult to handle in 3D vs 2D. The point is that it should be “reasonably” possible for someone to make a 3D scene from your scripting language, even though you are demonstrating a 2D scene for this project.).

Details

Your script should handle the following motions/behaviours/whatever:

* **Command “Move To”** : waypoint - based on **location** *or* **time**. For example: “move to location xyz in w seconds”, or “move to location xyz at max speed V” (note the 2nd one does not specify the time)
* **Command “Orient To”** : like, “Move To”, but for rotation/orientation
* **Command “Follow ‘curve’”** : Two options here:
  + A series of “waypoints”, essentially a series of “Move To” commands, or
  + Something like a mathematical curve (like a Bezier curve) where you specify a series of “control points” (BONUS). Note that this has to be someone complex – a Bezier curve or following a circle/ellipse would be the *minimum* here (you *can’t* say “follow a line” for instance)
* **Command “Follow Object”** :
  + Additional specifications:
    - follow distance (the closest we’ll get to the object)
    - follow “offset” (is object “2 units above” another object”, for instance.
    - Max follow speed (min speed is zero)
    - Acceleration/deceleration range: These are the “slow down” and “speed up” distance ranges away from the object, so beyond a certain “far” distance the object is moving a “max follow speed”, when it is a certain “near” distance from the object, it is not moving – between this range, the speed is proportional to the distance: in other words, the object will “slow down” as it gets closer to another object, rather than abruptly (instantly) stopping.
* **“Location based Trigger”:** When an object enters a certain area (defined by an AABB or Sphere), a command/command group is executed
* **Command Groups**: of two types
  + A “**serial**” set, where one command completes, then leads to another (eg: “*Move here in 9 seconds*”, then “*turn 90 degrees in 4 seconds*”, then “*Move here...*”, etc.),
  + A “**parallel**” set, where multiple commands are executed at the same time (eg: “*Move here in 9 seconds*” WHILE AT THE SAME TIME “*turning 90 degrees in 4 seconds*”, etc.
  + These command lists can be for one object (i.e. the animation for a specific, single, model), or multiple objects at the same time
  + These can be chained together as well, so there could be a set of “serial” commands that are themselves a set of “parallel” commands, and vice versa. SUGGESTION: force all “commands” to actually run inside “command groups”, and only schedule the “command groups”: if you have a set of sequential commands for a single object, then you would simply add these commands to chained together individual commands
  + Option : Given a set of “serial” commands which are groups of “parallel” commands which are controlling different objects, you can choose:
    - To wait until the entire parallel group is done before you move to the next serial command (i.e. wait for the “longest” command, time wise, to complete), or
    - (BONUS) Independently schedule the objects – i.e. any sequence of serial commands related to the same specific object are run independently of others. Note: to get this mark, your script would have to add a set of serial commands consisting of parallel commands relating to different objects – you can’t just create a set of independent serial commands (although you could have this happen in the background when the script is run to handle this, though... nudge, nudge, wink, wink)
  + For “serial” sets only: a “repeat” value: either a number, indicating how may times to repeat (0 is *don’t* repeat/execute once), or a symbol meaning “infinite”, meaning repeat forever.
* **Two (2x) “Special” Commands**: These are “non movement” types of commands that will call one or more functions/methods on your game objects/entities/actors.
  + Here’s some examples to try and explain what I mean (some of them are silly): Self-destruct; fire missile; turn on headlights; give birth to another entity; open landing bay doors; retract landing gear; etc., etc.

If you want a clear (ish) explanation of this, check out the FIRST Robotics “command based” manual (keep in mind this is for *high school students*) here: “Command Groups — FIRST Robotics Competition documentation” (<https://docs.wpilib.org/en/2021/docs/software/old-commandbased/basics/what-is-command-based.html> ) and here: “What Is “Command-Based” Programming- — FIRST Robotics Competition documentation”

Note that you *don't* **have** to implement this FIRST Robotics system, but it is a sensible, proven sort of structure that I'm suggesting might be a helpful design, or at least a starting point.

Lua script requirements:

In a theoretical "perfect world where everything is data driven", all aspects of your objects could be completely controlled by Lua script.

However, that's not practical or likely not even desired. In addition, considering that Lua is essentially a weakly typed, managed, scripting language, and C & C++ aren't, there's going to be issues interacting between the two languages.

Also, since Lua is interpreted and so very much slower than compiled code, there's certain aspect that would be pointless to implement in Lua script (like collision detection, as one example).

I'm thinking there's three main ways that Lua can be used:

* As part of the "Update()" method on your object (the one that's called every frame to update the object state). Each object could have a Lua interpreter instance (state) and this Update() essentially runs one or more Lua script(s) each time Update() is called. There are "hooks" into the object, either directly (keep in mind that Lua calls *functions*, not *class instance methods*, so this might be trickier), or indirectly (through a "manager" type of class/object), to update the object.   
    
  In a way, this is the more "powerful" option, in that there is 100% control over the Lua script for each object. On the other hand, having the script associated with a *specific* object makes interacting with other objects potentially trickier.   
    
  There is also the issue that the Lua scripts are only run once every Update(), so you would have pass the state each time, etc.   
    
  Using this method, you wouldn't even need the "control system", as suggested with the FIRST Robotics method, but could implement all the behaviour in Lua.
* As part of a centralized control system, running one "master" Lua script, which controls the behaviour of all the objects, or as a set of scripts, but still running on one centralized object, like a "world" object. The Lua script would be more complicated as it's managing all objects, but having interaction between objects would be simplified (the central controlling object "knows" about all the objects).
* The script runs either on a centralized "master" object or as part of the "Initialization()" call on the individual objects, but only "parses" the commands and "dispatches" the actual behaviour objects (the FIRST Robotics command objects). In other words, the Lua code is "setting up" the actual scheduling objects, running once at the start, then not running until another "command" is needed.

Again, these are only suggestions and you are free to do anything you want. However, the Lua script component has to be a key aspect of the application, and if not, you will lose a substantial number of marks. *Basically, if you don't incorporate Lua at all, you can't pass this project.*   
  
Also, if you do something to "game" this requirement (<https://en.wikipedia.org/wiki/Gaming_the_system>), by doing something like: adding an empty Lua script to the objects, have a "print("do everything")" command and that's it, then it will be treated as having not incorporated Lua at all.

**Again, I'm looking for the ability to control your animations *entirely* through the Lua script(s) without the need to change any code or without any "custom" script code that are running at the same time. In order to change the animation of your application, you should only need to update Lua script(s) and nothing else.**

**Additional parameters/aspects that all commands/command groups need:**

* A “friendly name”, “type”, and unique ID.
  + The **unique ID** can be automatically generated (I’d suggest you do that)
  + The **“type”** is for your use, for things like a group of commands. For instance, you might have a set of serial commands called “**Patrol\_Area**” that just moves a bunch of objects around the scene, forever (i.e. the serial command group “repeat” is set to infinite), or a command like “**Run\_out\_the\_guns**”, which animates the “arming” sequence of a battleship for instance).
  + The **“friendly name”** is used for naming the command and maybe for “calling” a command and/or adding them to a command group. These *don’t* have to be unique (I’d suggest that they are), but if they aren’t, *and you’re using them to schedule commands*, you need to decide what happens if the “scheduler” finds a number of the same commands: Run the first one? Run all of them? Run them in order (and what order – the way they are listed in the file?)?
* Ease in/out, which is an acceleration/deceleration (time, distance, start vel, end vel), which is what actual, physical objects have (cars don’t *instantly* go from 0 to 30 kph for instance): Note that your commands wouldn’t have all four of these options, as it depends on the type of command. For instance:
  + “Move to XYZ in 10 seconds” could have either:
    - **Time:** Ease in for 1 second (start of command), Ease out for 2 seconds (at the end): This would mean that the object would accelerate/decelerate from the starting speed to the ending speed over the first 1 second, then do the same at the end
    - **Distance:** Ease in for 10% distance (start of command), Ease out for 15% distance (at the end): This would mean the object would accelerate/decelerate from the starting speed to the ending speed over the first 10% of the distance, then accelerate/decelerate during the last 15% of the distance
  + **Note:** I’m *not* looking for an *exhaustive* list of every single possible combination of these, but that you have implemented a “reasonable” subset. What I’m saying is this:
    - Don’t stress yourself out, worrying that you’ve missed a couple variations, even though you’ve got a whole bunch of them
    - On the other hand, don’t test my patience with “*oh, I’ve got these two options, so I’m good, right? Heh-heh...I mean, they are the best options, right? Right???*” Remember that the intent is that there’s a bunch of non-programmers who are going to look at this “script” and think: “I want to do \_\_\_\_\_”, and there should be a range of options for them
* The script should handle the camera, too (i.e. the “camera” is really another object.
  + Note: this might mean that you have to implement a “look at this” sub command/option that only applies to the camera. You can either have option to have the camera look at anything (a location, another object, etc.), or just that if the object in question is the camera, that it automatically looks at whatever it’s looking at

Dire warning: Note that these “additional” items can “make or break” your mark for individual items. i.e. if you don’t have them, you can potentially lose all your marks if you are missing them. For instance, if your “move” command is something like “Move everything from xyz to xyz in 5 seconds” and it moves all the objects immediately, then stops, you’ll likely get zero for that (you’ll also likely to lose all the marks for the command groups, too, but that’s another story).

If this were, say, week 4 of term 1, it might be a different story, but at this point, we’ve covered a tremendous number of topics relating to discrete animation and control: various types of timers, physics and timing integration (different types, even!), acceleration curves, sine curves, cubic curves, LERPS, quatererions and SLERPs, and so on. You’ve got a giant “tool box” of code and techniques to handle the nuances of controlling movement and animation.

The “leeway” (-25% to -100%) is an attempt to prevent students “lawyering” me with technical details of the marking scheme – you know, when both the student and I are *clearly* aware that the student didn’t actually implement what they are asked (the “spirit” of the specification), but “technically” did *some* part, so should get “part marks”. My response to that is: Would you pay your plumber for “almost” fixing your toilet? What if your plumber was arguing that they “you should pay 75% of the bill because it’s 75% working” – you can’t actually use the toilet, but *another 25%* and you’d be good to go (pun intended).   
  
More specifically, it is an attempt to assess the range of options available for an animator. Even something like humble PowerPoint (and I’m talking like the early 2000s era PowerPoint) had a range of both animations as well as “tweaks” for each animation, available for use. I’m not expecting you to re-create PowerPoint, but have a look at what they have to offer, in the light of “I want to make a game cut scene”.

A jarring 0-MaxSpeed in 0 seconds, then moving, then instantly stopping is not going to get you any marks. You need to have *some* type of ‘ease in, ease out’. The additional marks relate to adding subtle control:

* Can you control the acceleration/deceleration?
* How many *ways* can you control this (Over a time interval? Over a start-stop speed?)
* Are there ways to combine different aspects? For instance, can a ship fly in, bank into a turn, while decelerating, then “level off” while it starts to descend? All of this at the same time? The point here is that there is a *large* range of control here, not just “one move command to rule them all” – with only one option (This is where the design of your system comes into play – it could be that *one command* can’t do all this, but perhaps with *a set of commands*, you can easily achieve these nuanced effects).

Something like only one type of move with one type (say fixed) ease-in/ease out, will likely earn you 25%.

Something like the “ship landing” sequence above (i.e. the *ability* to *script* this) would give you 100%. This is the ideal sort of situation that I'm hoping you can implement - the ability for an animator to "mix and match" a sequence of script commands to do a typical in-game cut scene/small movie, without having to call you in to re-write parts of the C/C++ code.

Something in between, in terms of scripting/ease-in-out/command-combination options would give a mark between those two.

Additional requirements:

You also *cannot* simply use an existing game engine (or part of a game engine, Ogre3D for instance, or implement this in something like Unity) to complete this assignment; it should be either completely new of significantly modified.

**You must include a video explaining how your script works**, giving examples. You need to show me where in your code the various parts are handled. In other words, you are “proving” to me that you have the various parts, rather than me driven to madness and frustration trying to see if you do (or don’t) actually have this stuff implemented; in other words: I’m going to assume that you **don’t** have the functionality if you don’t point it out (even if you point it out in your video, you could still be “talking out your \_\_\_” and *not* actually have the functionality as specified in the project. The video is really to help me understand your framework and how/where you’ve got your stuff.

(It’s also to help you get a bunch of demo videos ‘showing off’ your awesome code, yo!)

Grading Scheme

1. While you may freely “borrow” mine (or anyone other) code but your code should be “sufficiently” different from mine. See the "plagiarism" test, later in this document, for more details.
2. Further, you cannot simply use an existing game engine (or part of a game engine), even if it's a "from source" engine (i.e. you have the entire source) to complete this assignment; it should be either completely new of significantly modified. This includes, but is not limited to: Unity, Unreal, Cry, Anarchy, XNA, Cocos, Ogre, the framework from the OpenGL text, etc. In other words, you are expected to have made the vast majority (essentially all) of the engine in this term by yourselves, from "scratch" - i.e. starting from something a rudimentary as the "OpenGL Book" code or the GLFW starter code (we started with that in class).
3. If you code does not even compile, I will not mark it. Period.
4. While you may use some C++ 11 features, you are strongly encouraged to limit your code to the C++98/2003 standard.
5. You also may **not** use the boost library.
6. If you code does not build (i.e. linker error) and run (i.e. no crazy run-time crash that is unexpected), I may investigate this further, but only if there is some simple problem and/or slight configuration error.

Project Corrections

If any corrections or changes are necessary they will be posted to the course web site and you will be notified of any changes in class. It is your responsibility to check the site periodically for changes to the project. Additional resources relating to the project may also be posted.

75/10-year old “squinty eye” plagiarism test:

I have very little tolerance for plagiarism, but many students are unclear about what it is.

Basically, it’s submitting somebody else’s work as your own.

There is sometimes some confusion over this because you could argue nothing is actually “unique” (see: <http://everythingisaremix.info/> for a fascinating overview of this).

The whole point of assignments/tests/projects in this course (or any course, really) is to try to see if you are actually able to ***do*** the coding that’s asked of you. In other words: How competent are you? Handing me someone else’s code and/or making a trivial change isn’t good enough.

Also, it’s illegal (in Canada, the US, and Europe):

* <http://www.plagiarism.org/ask-the-experts/faq/>
* <http://definitions.uslegal.com/p/plagiarism/>
* <http://en.wikipedia.org/wiki/Plagiarism>
* <https://www.legalzoom.com/articles/plagiarism-what-is-it-exactly>

In other words, I’m not going to be drawn into a giant debate over how “different” your code is from mine or anyone else’s, if any sensible person (including me) would conclude that the code/application is pretty much the same thing, then it is. It is up to my discretion to decide this.

* While you may freely “borrow” mine (or anyone other) code ***but*** your code should be “sufficiently” different from mine (you might want to replace the word “sufficiently” with “significantly”).
* In other words, you *cannot* simply use an existing game engine (or part of a game engine) to complete this assignment; it should be either completely new of **significantly** modified.
* How will I determine this?
  + If I showed your application and/or your source code to either a pragmatic 75-year-old mother, or a typical 10-year-old, or even some random person walking down the hallway (i.e. a non-expert), and they looked at it, tilted their heads, squinted their eyes, and said “you know, they look the same,” then they ***are*** the same.
  + Another test would: How much time it would take for a "competent programmer" (for example, *me*) to make the changes you are submitting? The point here is that I don’t “care” if you tell me “But it took me *weeks* to make the changes!” Fine, but if I can make those same changes in 10 minutes, then not a lot of work has been done (certainly **not** sufficient work – these projects should show take **days** of work having been done).

Possible Groups

You have a choice of being in the group (of 2, so really, “pairs”) or not.

If you choose to be in a group, all group members receive the same mark.

Keep in mind that you will be doing the exams (and interviews, and a lot of your job...) alone…

I am not going to get involved with issues like “I did all the work, and he/she is a lazy bum”, or “we didn’t know/verify what so-and-so was doing, and when I checked, that had done nothing/everything/changed everything/moved to Florida/ate the flash drive with our only copy”, and so on; if you choose to work in a group, **you also choose to manage the dynamics of the group entirely yourselves**. This also includes group dynamics taking place in other courses that you feel are impacting this course and/or the policies/practices of other instructors regarding groups.

The only exception to this is something that would be considered an “exceptional circumstance” (as defined by the Fanshawe Evaluations policy), which practically means: accident, illness, death, etc.

Note that even under the policy, this is a “grey area” *when it comes to projects*, as, unlike a test/exam, you’ve had a relatively long time to work on it – if someone gets sick on the due date, *and you have absolutely nothing to submit*, that’s not quite in the spirit of an “exceptional circumstance”; it’s more like you did nothing until the last minute and fate/bad luck intervened.

You must indicate your grouping ***before*** submission. This can be in the comments portion on the submission page of Fanshawe Online (I would prefer that, actually, rather than a separate e-mail). What I will *not* accept is situations where after the assignment is submitted and marked, you try and tell me, with a shifty looking expression: “Yeah, I was in Bob’s group. Yeah, that’s it.”, or give me ETOTIs (“Epic Tales of Tragedy an Intrigue”, see earlier paragraph).

If there is *any* debate/confusion/disagreement about people being or not being in a group - i.e. one person claims to be in a group, yet the other person claims they are *not* in a group – then you are *not* in a group, and your submission (or lack of submission) will be treated accordingly (also, if you both submit the same thing, that’s cheating/plagiarism).

Only one student needs to submit, but if you are a realist (like me), then you may also have every member of group submit the same project to FOL. However, you need to make sure you are all submitting the same (i.e. “current” or “latest”) version of the code: if everyone in the group is submitting slightly different code, I won’t look too kindly on having to spend a ton of time determining what parts are “different” and/or “correct” and/or “the latest”

Either place this grouping information (or any other note you expect me to read) in the FOL comments, in an e-mail, or in some *obvious* place, like a “readme” file on the root of your project folder. Please don’t place them deep inside the comments of one of the files, buried within the code somewhere, in some alternate dimension, carved under a tree in some forest somewhere, etc.