|  |
| --- |
| INFO6044 Game Engine Frameworks & Patterns  Project #1 Fall 2023 |
| Weight: Project weight evenly distributed across 2 to 3 projects. |
| Due Date: Let’s try: Tuesday, October 31st, 2023 @ 11:59 PM |

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

Description and Purpose

To demonstrate your understanding of commonly used design patterns and technique, you are to recreate a classic (i.e. super simple) game from the 1980s in C++.





To clarify, you don’t have to implement *everything* in Robotron: 2084, but the main gameplay and action.

Some comment about this being “too difficult”:

* The original game was developed in six months by two people...  
  BUT
* They also developed the hardware and had no graphics API.
* The tools they used can’t even be compared to the modern tools you have.
* Most of it was written in assembly language – they had to deal with the hardware directly.

Using the new:

We’ve looked at a number of concepts up until now:

* “Stack” based vs. “heap” based memory allocation:
  + Using “new” 🡪 heap based
  + Not using “new” 🡪 stack based
  + ...and all the trade-offs between the two ways
* Public, protected, and private inheritance
* Interface classes (specifically “pure virtual” interface as used in C++)
* Virtual (polymorphic) and no-virtual (override) types of inheritance
* Some software design patterns:
  + Abstract Factory pattern
  + Builder pattern
  + Mediator pattern
  + Singleton
  + Pointer to Implementation (PIMPL pattern)
* Limiting the impact of extending/changing your application (you could say the reason for the design patterns, and this entire course, really, is to assist in this way) Specifically, it's the ability to substantially add/change features and functionality with minimal impact on the build - ideally only causing the specific parts you changed to be changed/updated. Your changes shouldn’t impact unexpected parts of the code.

While you are free to implement any design you’d like, you will get marks for using some of software design patterns/techniques we’ve looked at.

Details

In the “Robotron 2084” 7zip/zip file, there are a number of resources you can use:

* Robotron 2084 - sprite map is a ripped bitmap scan of the actual graphics (for a more modern PC port, but they are the same)
* robotron - 2084 manual.pdf is a game manual for the player, explaining what each character is, how they behave, etc.
* A few links on how the original game was developed (just for interest)
* A link to a game jam entry where someone made a variant of the game in a couple days.
* A folder called “Sprite map” which you can use to make your own characters (though I will give you a tool to do this if you want to use that).

Marks

**This is in two main parts: what you implemented in the game and what techniques you used to do it.**

**Game and game-play:**

* *Marks are given on how closely they match the original game or a “reasonably good” port of the game: see the “robotron – 2084 manual.pdf” for behaviour descriptions.)*
* *No “Animations” of the objects = 65% of the mark.*
* *Objects are to mimic “pixel” art, so I’d suggest using cubes for each “pixel” of the original game*

|  |  |
| --- | --- |
| **Item** | **Mark** |
| Playing area with a border | 0 (no marks for this) |
| Main character/player moves in eight (8) directions | 20 |
| Main character/player shoots in eight (8) directions | 20 |
| Objects player shoots are destroyed | 10 |
| Grunts implemented (and kill player) | 20 |
| Humans walk around | 20 |
| Humans can get rescued | 10 |
| Indestructible hulk implemented (and kill player) | 20 |
| Spheroids implements (and kill player) | 15 |
| Enforcers and tracking projectile implemented (and kills player) | 30 |
| Brains and tracking missiles implemented (and kills player) | 30 |
| Humans converted to “Progs” when touched | 15 |
| Tanks and bouncing orb projectiles are implemented (and kill player) | 30 |
| BONUS: Score is implemented (as sprite/objects) (10%) |  |
| BONUS: New levels start after old levels end (5%) |  |

**The patterns/techniques: This “scales” the mark you get above. i.e. the mark above is multiplied by this mark to get your final mark.**

|  |  |
| --- | --- |
| **Item** | **Mark** |
| Abstract Factory utilized | 20% |
| Builder pattern utilized | 25% |
| Mediator pattern utilized | 25% |
| PIMPL (pointer to implementation) utilized | 25% |
| Interface pattern (in C++) utilized | 5% |

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. 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.

I am not going to get involved with interpersonal issues; 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.

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). Make this absolutely clear (not buried in some comment somewhere).

If there is any debate 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 you may also have everyone submit the ***same*** project independently.