|  |
| --- |
| INFO6044 Game Engine Frameworks & Patterns  Project #1 Fall 2024 |
| Weight: 50% of the “projects portion” or 30% overall. |
| Due Date: Saturday, November 2nd, 2024 @ 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 **one** of these classic (i.e. “pixelated”) games from the 1980s with C, C++, and OpenGL:

|  |  |
| --- | --- |
| **Galaga (1981):** | |
| GitHub - opengateware/arcade-galaga: Namco Galaga Compatible Gateware IP  Core for FPGA | 🎮🕹️👉Galaga (1981) - Gameplay Arcade |
| **Donkey Kong (1981):** | |
| Donkey Kong (game) | A history of the Mushroom Kingdom Wiki | Fandom | Donkey Kong Original Edition |
| **Defender (1981):** | |
| Defender | 1981 | Arcade | Gameplay | HD 720p 60FPS | Planetoid - Defender fan game - Dev diaries - Defold game engine forum |

Amazingly, these games were all released in 1981.

It’s also interesting to realize that releasing a game at that time often involved developing (or helping with the development of) the circuit boards, etc.

[](https://www.ebay.ca/itm/166920611654?_skw=galaga+arcade+pcb&itmmeta=01J9KKJY6DT0SFRATZD438TXSE&hash=item26dd3e6746:g:vqAAAOSwMbBmDHBS&itmprp=enc%3AAQAJAAAA0HoV3kP08IDx%2BKZ9MfhVJKmqiapL2QZd8BI%2BH9vKpIcDQ0FXzsS0KlX7aNvqv5%2BfaPtR56apZAW37RzSAkTYD%2BWb0zk%2BejoozitK1zcDOJdwcHHQA87smZNyxPyXT06QXRP0XLo%2BtWIc8DCy6KAYWDnA2dXSqE8m%2FsticEp2THavTF9lbAwQkeJBclfCKwMiASgTGYIDfhIJH78r%2BXWx5vlK5BGs57l1rgsVhJRHKvUqwLS7uVZxTv5Phnum7vLsfr7BzBJvqFctozzVlUgb--I%3D%7Ctkp%3ABk9SR7bjy_PMZA)There also wasn’t any “operating system” to speak of, exceptionally limited tools, and the hardware these were running on was almost comical compared to now. Like your microwave oven 100x more processing power, likely.

Here’s an actual “Galaga” main board that you can buy on eBay today!

You don’t have to implement *everything* in these games, mainly just the things on the 1st level.

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 and they had to deal with the hardware directly.

Details

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
* Abstract Factory 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 designs/techniques you’d like, part of your marks will be how you set up your design to be maintainable and extensible**: i.e. all of these games had squeals or were sequels to other games:

* Galaga is a sequel to Galaxian (1979):
  + Galaga aliens have a tractor beam that can capture your ship. When you destroy the capturing ship, you can use the freed ship for extra firepower.
  + Galaga had a series of “challenge levels” where patterns of ships would attack in predictable waves.
* Donkey Kong Jr (1982) is sequel to Donkey Kong:
  + Now Mario is the villain, and you play as Donkey Kong Jr., rescuing your Dad (Donkey Kong)
  + You can jump left and right to climb up and down vines.
  + There are multiple enemies, some of which can drop things on Donkey Kong Jr.
  + You can knock things which wall on enemies.
* Stargate (later in 1981) is a sequel to Defender (1981):
  + Many more enemies with varied attacks and different modes.
  + Some of these enemies drop other weapons (bombs/mines) and/or explode/transform into multiple different enemies.
  + If all the humanoids (the things you are trying to rescue in both games) die, the entire planet explodes, and every lander becomes a Mutant (the lander + abducted human).

One key pattern here is that the player and/or enemies have new variations to them, along with additional ways of interacting.

Keep in mind that these squeals came out very quickly after the original games.

|  |
| --- |
| **What I’m looking for is you explaining how the features and functionality of your game could be easily and quickly done. In other words, your clever implementation *could* handle adding additional enemies, behaviours, interactions, etc. without mostly or completely re-writing it.** |

3D Pixelated models:

The output of your game needs to look “pixelated” but using 3D cubes for each pixel, like this:

The **DonkeyKong\_Level\_0\_with\_items.ply** file is shown to the right to give you a sense of what I’m looking for: Note that the Z scaled to 5x thicker than the file to accentuate the 3D effect.

**You may NOT use textured sprite maps and you MUST use a perspective projection transform (as opposed to an orthographic projection); they must be 3D cubes;** if you moved the camera location or changed the field of view, you should be able to see this 3D effect.

If you are unsure about the projection, we’ve only used a perspective one – we’ve never used an orthographic projection.

There are a couple helper application you can optinoally use to “transform” the various sprite maps into a individual 3D models:

* Robotron3DTextToModelThingy: this is from last year’s group and uses a text file to generate the 3D cube models. There’s also an excel file that I used to set this up.
  + I took each sprite map and manually entered them into the Excel document, marking each colour with a specific letter. Then I copied the regions in the Excel document, saved it as a text file, then feed it to the Robotron3DTextToModelThingy program, generating the files.
* **SpritePNGtoCubePixelHelper**: new for this year (and likely far more helpful), this takes a PNG file and generates a 3D ply model as output.
  + There are a few command line options that help, like ignoring the background colour and cropping out extra background.
  + You can see these options when you run the application, or see various “.bat” files (for example: Generate\_DonkeyKong\_Models.bat) for the settings.

Feel free to use these helper tools and/or the 3D models that it output.

The sprites I found online are under the SpritePNGtoCubePixelHelper/textures folder. Under that folder are the three games and a “Sprites (isolated)” folder where I’ve cropped the individual sprites.

Under SpritePNGtoCubePixelHelper/PLY\_Models is a “.bat” (batch) file that will (should? 🤷‍♂️) generate all the 3D cube model files for each sprite (though you will have to build the Release version of the SpritePNGtoCubePixelHelper program to have the .bat file work properly. There is also a .7z file with all the models that I initially generated (Defender\_Original\_Generated\_Files.7z for example).

Alternatively, you can generate these levels sort of like the original developers and create an output array of “pixels” each being a 3D cube. From this, you could make functions to draw lines or make “sprites” of these 3D cubes. For instance, the “ground” of the defender game was almost certainly created by drawing lines of pixels at run-time; you could do the same sort of thing, but “draw” a “line” of 3D cubes.

If you are curious how to read the original PNG files, you can see that from the <https://lodev.org/lodepng/> site (specifically example\_decode.cpp) or the SpritePNGtoCubePixelHelper source code.

What you are to submit:

* Your entire solution file with everything I need to click on “build” and then “run”.  
    
  (PLEASE for the love off all that’s good in the world: REMOVE any temporary garbage I don’t need, like the .vs folder, etc.)
* A **video** of you *demonstrating* your solution with your voice explaining what I’m seeing.   
  Again, I’m not looking for some Oscar winning piece of cinematic mastery – imagine you are demonstrating it to me.
* You will also be demonstrating it to me *in person*, so I can ask questions about it. This will either be done in class or during office hours, or we can schedule a time.

Marks

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

**Your technical decisions and techniques:**

The patterns/techniques: This “scales” the mark you get below. i.e. the mark you get for the game & gameplay (out of 100%) is multiplied by this mark to get your final mark.

|  |  |
| --- | --- |
| **Item (you would explain this in the video and in person)** | **Mark** |
| How entities (likely enemies) are “extended” to add functionality from one to another, or not… (for instance, the Mutants *could* be extended from the Lander enemy in Defender. I’m looking for examples of where you did this, and how, or why you *didn’t* do that and what you did instead. | 25% |
| Give me **at least two (2) examples** of where your initial approach *didn’t* work while you were adding functionality and what you did to solve this. | 25% |
| Abstract Factory and/or “interface” pattern was utilized, or why you didn’t do that. Note that if you didn’t do that, it can’t be something like “I didn’t know” or “I don’t know how” or something – I’m looking for a *technical* pro/con cost/benefit breakdown of why you made that decision. | 25% |
| Give me at **least two (2) examples** of how you could enhance/update the game – like if there was a sequel to the game. I’m looking for things like “Mario could get another weapon besides the sword” or “The Galaga ship could have a tracking missile, too”: then you’d explain how your very clever code would easily handle this kind of update *without* reworking the entire thing. (Note: This must be something that would *reasonably* be an improvement/enhancement, like something that a 12 year old would say "Oh Defender 2 is *way* better than Defender because \_\_\_\_\_") | 25% |

**Game and gameplay:**

Marks are given on how closely they match the original game or a “reasonably good” port/emulation.

|  |  |
| --- | --- |
| **Galaga** (https://www.youtube.com/watch?v=dvjapcHsqXY) |  |
| **Item** | **Mark** |
| Rendering is 3D where each pixel in the original game now looks like a 3D cube. | 10 |
| Player ship moves horizontally along bottom of screen | 10 |
| Player shoots projectiles towards the top of the screen. | 10 |
| Player can have multiple "shots" moving at the same time (i.e. not just one shot at a time) | 10 |
| Stars are moving down in the background | 10 |
| There are bee and butterfly aliens (bees have yellow wings, butterflies have red wings) | 10 |
| Bee and utterfly aliens are destroyed with one hit | 10 |
| There are moth aliens (with green wings) | 10 |
| Moth aliens change blue when hit once | 10 |
| Blue moth aliens (green ones with one hit) are destroyed with 2nd hit | 10 |
| Aliens and player have an "explosion" animations | 10 |
| Aliens attack along curved lines | 10 |
| Aliens shoot/drop projectiles at the player that are visibly distinctive from the player's | 10 |
| Player ship destroyed with one hit from alien projectile | 10 |
| **Total:** | **140** |
| BONUS: Enemies appear in waves, following curves to end up in their starting grid formation (rather than the entire formation appearing at once) | 5.0% |
| BONUS: Score is implemented (as sprite/objects) | 2.5% |
| BONUS: Player has multiple ships and loses one each time their ship is destroyed | 2.5% |
| BONUS: New levels start after old levels end | 2.5% |
| BONUS: Challenge stages exist, including a bonus score for destroying all the aliens in an attack | 5.0% |
| BONUS: Bonus summary screen shown. | 2.5% |
| BONUS: Moth aliens "tractor beam" ship capture is implemented: tractor beam captures ship; shooting alien with captured ship will release ship; ship attaches to your original ship giving multiple shots at a time | 5.0% |
| BONUS: Scorpion **and** (green) manta ray **and** dragon fly ships present in later levels (3 additional types of aliens). They need to appear like they do in the later level, often coming in from the sides or appearing behind other aliens | 5.0% |

|  |  |
| --- | --- |
| **Donkey Kong** (https://www.youtube.com/watch?v=KJkcNP4VkiM) |  |
| **Item** | **Mark** |
| Rendering is 3D where each pixel in the original game now looks like a 3D cube. | 10 |
| Mario moves left and right along the platform | 10 |
| Mario climbs ladders "correctly" (can only go up and down when on the ladder) | 10 |
| Mario can jump | 10 |
| Barrels "roll" down the platforms | 10 |
| Barrels "roll" can roll down the ladders | 10 |
| Bright blue barrel starts fire in oil barrel | 10 |
| Flame enemies move along platforms and up ladders | 10 |
| Barrels and flame enemies "kill" can kill Mario | 10 |
| Mario can get and use a hammer and hammer destroys barrels | 10 |
| Score shows when jumping over barrels and hit with hammer | 10 |
| **Total:** | **110** |
| BONUS: Has startup animation where Kong jumps and the platforms "fall" to final location | 5.0% |
| BONUS: Barrel destroying animation present | 5.0% |
| BONUS: Has "HOW HIGH CAN YOU GET?" intro level | 2.5% |
| BONUS: Count down "bonus" score present | 2.5% |
| BONUS: "Kong grabs princess and climbs up" end of level animation present | 5.0% |
| BONUS: Level 2 level present with: \* Bonus items present (umbrella, purse, etc.) \* Rivets are present (the things you walk over to remove) \* Flame enemies present (can kill with hammer, etc.) | 10.0% |
| BONUS: Level 2 - end of level platform collapse present | 2.5% |
| BONUS: Level 3 (pogo-stick/spring enemy) level present: \* Has "elevator" platforms \* Bonus items present (umbrella, purse, etc.) \* Spring and flame enemies present | 10.0% |

|  |  |
| --- | --- |
| **Defender** (https://www.youtube.com/watch?v=wAKxa5C9jHY) |  |
| **Item** | **Mark** |
| Rendering is 3D where each pixel in the original game now looks like a 3D cube. | 10 |
| Has scrollable "ground" and stars which moves left and right when player moves | 10 |
| Player ship can go up and down, thrust "forward", and change direction | 10 |
| Player ship location scrolls to correct side when direction changes. | 10 |
| Player shoots multiple large projectiles, with "trails" that destroy enemies and Astronauts | 10 |
| Astronauts appear and gently fall to the ground | 10 |
| Landers grab Astronauts and bring them to top of screen | 10 |
| Landers + humanoid at top of screen become Mutants | 10 |
| Player ship can "catch" falling Astronauts and place them on the ground | 10 |
| Mutants more aggressively attack players | 10 |
| Baiter enemy (looks like a classic 1950's movie UFO) present | 10 |
| Player has three "smart bombs" which destroys all enemies within range (on screen) | 10 |
| Player has "hyperspace" button which teleports to a random location | 10 |
| Mini-map is present at top showing entire planet, entity locations, and the visible area | 10 |
| Total: | **140** |
| BONUS: Bomber enemy present | 5.0% |
| BONUS: Pods and Swarmer enemies present | 5.0% |
| BONUS: When all the Astronauts die, the planet explodes, and all remaining landers become Mutants | 2.5% |
| BONUS: Planet reforms if all the Mutants are killed. | 2.5% |
| BONUS: Score, number of remaining ships, and bombs UI is present | 5.0% |

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.

“Show stopper” marks:

* You *aren’t* displaying “pixels” as 3D cubes with a perspective projection = **mark of zero (0).**  
  + No bitmapped sprites, for example.
  + No orthographic projection
* Normally a grade of zero will be assigned to any assignment that is submitted late. However, certain rare exceptions apply according to the Fanshawe policy for missed evaluations.
* If you code does not even compile and/or build, I will **not** mark it. Period. (which means mark = 0)
* I may also give up on marking (i.e. give a mark of 0) if there’s too many exasperating run-time issues. The odd/occasional crash isn’t a concern, but if I can’t even run it without it routinely crashing or doing something maddening, then I’m just not going to mark it. Beyond terrible application interfaces (I’m looking at you, Microsoft Teams…) this is supposed to be a game, where most people have exceptionally low tolerance for badly behaved software. So this includes insane mouse controls, virtually unusable mouse sensitivity, the mouse constantly being “on” even when the window isn’t in focus, crashing every time you alt-tab to another app or minimize, etc.

While these aren’t “worth” anything, these will get a huge penalty (possibly zero marks):

* The use of “auto” (mark of zero)
* The use of “boost” (mark of zero)
* The use of an 3d part library that hasn’t been cleared by me (glm, GLFW, and assimp are OK)  
  (likely mark of zero)
* Using the output window/context/canvas of ImGUI for your main output (mark of zero)
* No explanation video (mark of zero)

Project Corrections

If any corrections or changes are necessary, they will be posted to the course web site and github 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.

80/10-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.

This also extends to AI generated code. As far as I’m concerned, if an AI generated it, then you didn’t.

But AI is also very helpful for learning stuff, so the sentence above covers off the “plagiarism” part: if I suspect you didn’t create the code (and are just using it), then I’ll ask you about it and if it seems to me that you have no clue what the code is doing or why or how you might alter it, etc. than I don’t care where it came from because *you* clearly don’t and are just using someone/something else’s code.

Also, it’s illegal:

* <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.