# MI 231 2024.Fall – Classic Game Project – Assignment

**Final Due Date: Monday, November 11th (Skyrim Day), 2024 @ 10:20am (in class!)**

**Goals / Spec:**

Working in pairs, you will faithfully recreate core features of an arcade/console game of your choice (with approval) using C# inside the Unity game development engine. You will be responsible for recreating an agreed-upon "classic level"[1] of the original game in exacting detail. On the day of kickoff of the assignment, you will agree with the professor about which game you will make.

There are several goals to this assignment:

1. You will solidify your knowledge of C# and the Unity game engine (while not having to simultaneously design a new game from scratch).
2. You will get a chance to examine and recreate the esoteric design details of a gaming classic. There are dozens of tiny details in these games that made them rise above all of the other games of the era, and many of them are not immediately obvious.
3. You will gain experience with the Agile development methodology—including **burndown charts** and **daily scrums**—that will be used throughout the rest of the class.
4. You will take part in the iterative process of design that is the cornerstone to good game design.
5. You will get a chance to see how the game mechanics and technologies that you develop for the recreation of the classic level can be repurposed for your custom level.
6. **Remaking or reusing the original graphics is *not* a goal for this project**. You will be scored on how well your game emulates the feel of the original game but very little score will be given based on how good the graphics look.

**Additional Requirements:**

1. Your game will be delivered as **both a GitLab project and a Unity WebGL build uploaded to D2L.**
2. Your game will also be **presented in-class on the due date**. If working in pairs, both team members should be present to present your game.
3. You must **track the progress of your classic game project using a Burndown Chart** that we provide.
4. Again, please remember that the **feel and mechanics** of the game are ***drastically*** more important for your grading than the graphics.

**Keyboard Controls:**

**All games must use the following keyboard controls**, which map to the original NES controller:

- **Arrow Keys** and WASD – Up, Down, Left, and Right
    - If you use Input.GetAxis("Horizontal") and Input.GetAxis("Vertical"), this will happen automatically.
- **Z** – B on the original NES controller.

[1] A single level of a game would be something like World 1-1 in Super Mario Bros. or a single dungeon in The Legend of Zelda. If you have any question about what would be equivalent to a "level" in your game, please speak with Jeremy or your LA as soon as possible.

- **X** – A on the original NES controller.
- **Shift (make both shift keys work)** – Select on the NES controller
- **Return/Enter** – Start on the NES controller

If your game needs other controls, please contact us on Piazza about it within the first week of the project.

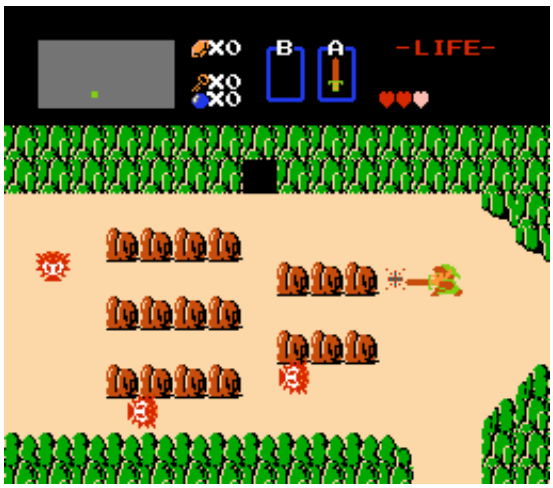
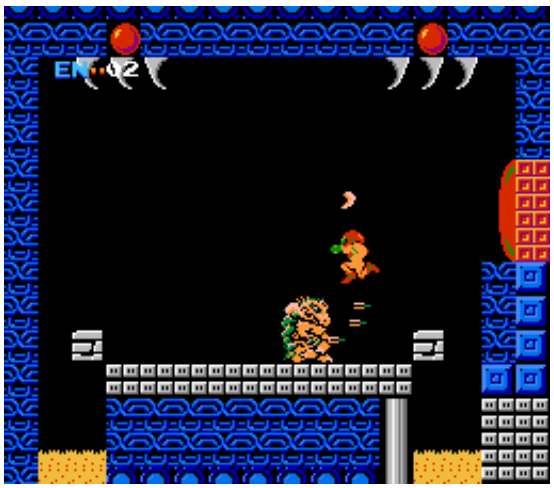
**Mindset: (This is actually much more important for the Final Game Project, but it's still something really good to keep in mind!)**

Read the Gamasutra article "How to Prototype a Game in 7 Days" by the Experimental Gameplay Project team from Carnegie Mellon (link also follows this paragraph). Then check out their website to see the kind of fantastically creative games that can be made by a single person in just one week if approached with the right mindset, time, and abilities: http://experimentalgameplay.com/ . Remember that in this project, it's much more important to have something that plays properly than something that is coded "properly". The key is capturing the feel of the game and the game mechanics, not fancy coding.

https://www.gamedeveloper.com/game-platforms/how-to-prototype-a-game-in-under-7-days

**Example Games ( You can see several example games in this spreadsheet )**

Here are two example games that you might choose (though you aren't actually allowed to do Zelda because of the Dungeon Delver project that you've already done).

| The Legend of Zelda (NES) | Metroid (NES) |
|---|---|
|  |  |

We are expecting different things from each different game.

**The Legend of Zelda** has sliding blocks, items that confer abilities, inventory management, and hidden rooms, but each room is only one screen in size.

**Metroid** has scrolling screens (in two directions), lots of changes to movement and abilities, but very simple enemies.

**Grading:**

You will be graded based on the following criteria (totaling to 100%). For all of these, just doing exactly what is required of you will tend to get you a B. If you want an A or higher, you need to go beyond the base requirements. The difficulty of the game you select (**as shown in this spreadsheet**) will be applied as a modifier to this grade, with 3/5 difficulty granting no change in grade, whereas 1/5 difficulty will lower the grade and 5/5 difficulty will increase it.

**5% – Game Idea Pitches** – You should post three classic game ideas you're interested in pursuing for this project to Piazza for the instructors to review and comment on. This part of the grade also includes communicating clearly with the instructors which game you've decided to work on. You must choose a valid game that is deemed acceptable by the instructors.

**10% – Set Up Burndown Chart & GitLab** – You need to have selected your game and your teammate and to have set up and shared your burndown chart & GitLab on time. **Due in Lab Wed 10/23 @ 10am**. Share with: <gameprof@msu.edu> and <gerlac32@msu.edu>

**5% - Initial Playable -** Here we are looking to see that the very basics of your game are working. Do you have the level imported? Is the main character moving pretty well? You must show this to instructors in class. **Due Mon 10/28 @ 10am**

**10% – Alpha Playable** – Is your game on-track? At this point, **there is only one week left in the project**. Does your alpha include the core movement mechanics of your game? Does it include at least one enemy type? Is the layout of your map complete? **You *must* show us these builds *in class*.** This is the perfect time to let us know what you need help with. **Again, you must show this build to instructors in class! Due Mon 11/4 @ 10am.**

**50% – Classic Game Project Final Turn-in** – Did you turn in your project on time? Did you complete all the required tasks on the burndown chart? Did you fulfill the requirement below? **Due Mon 11/11 @ 10am.**

**(25%) – Mechanical elements** – How faithfully did you recreate the mechanics of the original? Does the game feel complete and internally consistent in the same way as the original? Did you capture the "special sauce" that made the game a classic?

**Game Feel** – Does the game move and feel like the original? Sprites and actual game art will not count at all for this aspect of the grading, however the "feel" of the game will (see the book *Game Feel* by Steve Swink). Don't forget about music.

**(20%) – Technical elements** – Does the game work? If a game isn't playable, it's not a game. Memory leaks, glitches, and other bugs will count against you for this requirement.

**(5%) – Aesthetic elements – Graphics and Sound can only count for this part of the grade, so having *perfect graphics and sound will only earn you 5%* of the total project grade!!**

**20% – Process: Agile Methodology** – This will be the first project for which we will be requiring you to follow the Agile development methodology. You will be creating **burndown charts** that must be **updated daily**. Your final burndown charts at the end of the project will be graded and count for this part of the grade.

**Agile Progress Reports and Burndown Chart**

As described in the video "Intro to Agile Scrum in Under 10 Minutes" ( https://youtu.be/XU0llRltyFM ), a burndown chart is a fantastic tool for tracking progress on a project.

**Final Deliverable**

You must submit a WebGL build and ReadMe file to D2L. You also need to share your GitLab repo with us. Double check to ensure that you can unzip this file and play the game. ***Projects that are unplayable because of a bad submission (due to corruption, incomplete submission, or any other reason) will be counted late until they are playable***. Be certain to **name your folder before zipping**!

**Setting-Up Your Burndown Chart**

1. Log in to Google. These burndown charts require a Google account.
2. Go to the **Classic Game Burndown Chart Template**
3. Go to the burndown chart template for your chosen game and then choose File > Make a Copy… from the menu to make your own copy.
4. Name the new file F24-231-Classic-FirsnameLastname_FirsnameLastname-GameName (e.g. F24-231-Classic-JeremyBond_ElexMcCoy-Metroid) and click Ok. This will create a new copy of the template in your Google drive.
5. Share the new document with the LAs and me:
    1. Click [Share] in the top right corner.
    2. **Add <gameprof@msu.edu> and <gerlac32@msu.edu> in the Invite People text area.**
    3. Choose "Can Edit" from the popup to the right of that text area.
    4. Click [Share & save].

**Questions**

The class forum is available through Piazza and it is *by far* the best place to ask questions.

**The Classic Game Starter Kit**

If you're planning to make a platformer, you should definitely check out the **Classic Game Starter Kit** that I created for this class (based on the Platformer Toolkit made by the Game Makers Tool Kit YouTube channel but with many improvements and bug fixes).

**Best Practices for the Classic Game Project**

We've done this project *many* times, and these are some best practices that we've developed based on mistakes people have made in past semesters.

- Importing 2D Sprites – **Remember that making the game LOOK like the original game can only earn you 5% at MOST. Don't waste your time on this!!!**
    - Make sure that your sprites are imported with a Pixels Per Unit of 16. The reasons for this are covered at length in the Dungeon Delver chapter, but students still often forget.
    - Also make sure that any large images that you import are not being scaled-down by Unity on import. To do this, set the Max Size at the bottom of the Image Import Settings pane to 8192.

    - At the bottom of the import pane, you also need to set Compression to High Quality or None (which is even better).
    - If your sprites are blurry, set their filter mode to Point.
    - Great Resource for Sprites: https://www.spriters-resource.com/
- Don't scale anything! Especially don't scale anything non-uniformly. This has the potential to mess up all sorts of aspects of your game and physics.
- For WebGL builds specifically, audio needs to be in AAC format, and in the Resources folder.
- Don’t call any version of FindObject or GetComponent for the same object in Update. If you need to use the Rigidbody, get it at Start and store it instead of using GetComponent<Rigidbody>() every frame. Your game will run better.