Milestone 1  
Pitch, Presentation & Prototype

**Pitch / Presentation Due: March 5th, 2019 in class**

# Deliverables

* **Presentation** in class on **Tuesday, March 5th**
* Basic **documentation & timeline**
* Core Framework/Engine **code** (merge your individual assignments into a singular code base)
* Unity gameplay **prototype**

# Description

Form a team of approximately 3 – 4 students. Your first task is to decide on a simple, feasible game-like experience to create. Remember: The purpose of this game is to provide a context in which to implement interesting graphics concepts, not to flex your design muscles or create the most feature-rich game possible.

As a team, present a quick (5 - 8 minute) game pitch. Yes, this means use PowerPoint. Also make a very basic gameplay prototype of your game (see next page) and *begin* your core framework/engine

# Pitch / Presentation:

* The game itself
  + What is the core mechanic? (Scope down to this one thing!)
  + What makes it simple enough for your group to create this semester?
  + How will you handle art assets & UI?
    - Is your art style feasible for your group and the semester?
    - Maybe some *very pretty* boxes and spheres?
    - Feel free to show concept art, mock-ups or images of your inspiration
* Graphics programming
  + What kind of graphical effects are you planning?
    - Particle systems
    - Real-time shadows
    - Normal maps
    - Post-processing
    - Refraction effects
    - Etc. - This may require some “research” – play or watch some games!
  + Which parts of your game will be enhanced by the effects above?
  + Find screenshots or short videos of the effects you’d want to emulate
* Other technology? Don’t go too crazy here! More of this stuff often means less “GGP” stuff.
  + Alternate input devices
  + Physics – “real” or faked (or none)
  + Sound effects – fmod or OpenAL most likely
  + What else will you need to create this game? (Prototype should help determine this)

# Potential Game Ideas

Here are some basic game ideas that *could be* within the appropriate scope:

* Flappy bird
* 2048
* A *basic* infinite runner
* Shooting gallery (firing basic projectiles at simple shapes)
* On-rails shooter
* Air hockey

You don’t have to choose from this list! These are just examples.

# Completely Out-of-Scope Ideas

These ideas, while probably interesting, are going to be out of the scope of this project:

* Fighting game
* RTS
* RPG
* Procedurally generated dungeon crawler
* Racing game
* Physics-heavy platformer
* Open world game
* Anything that requires networking

# Documentation & Timeline

Provide a document containing your overall plan, including a week-by-week timeline of what tasks you think you have, who is in charge of each and what your plans are for when tasks take longer than you expect. You have until the end of the semester – how will you spend that time?

I expect this to be short write-up, but not just a list of bullet points. Put some thought into it. What other technology will you be using (external libraries, version control, controllers, etc.) All of this is important to figure out early on so the entire team in working towards the same goal.

Where do you expect to be at the end of the semester? Where will you begin? What happens between those two steps?

# Core Framework/Engine

Speaking of where to begin, you’ll also need to start coding your games. Since each group is making a different game, you’ll all have slightly (or majorly) different engines. You’re not expected to have everything in place at this point, but each group should have a singular code base that is able to do everything you’ve done in the individual assignments (and possibly more).

I don’t expect your C++ engine will be up to par with your Unity prototype at this stage, but it should be a good starting point from which to begin building.

# Prototype

Begin creating a basic *gameplay* prototype of your game. I’d suggest using Unity or a similar engine suited for rapid prototyping. Do not spend weeks and weeks crafting the perfect, feature-complete prototype. This should be a **basic**, game-mechanics-focused prototype to help illuminate the non-graphics features required by your engine.

I’m not expecting a prototype of every last detail in your game. The prototype should, however, allow the user to experience the **core mechanic** of your game. You don’t need to worry about art, special effects or shaders at this point either. Simply create some basic shapes to get a feel for the game (“grey box prototyping”). You should also use this opportunity to determine the kinds of algorithms and technology your game will need.

For example: you’re creating the prototype in Unity and you find that you’re using lots of rigid bodies & trigger colliders to get movement and collisions working. How do you plan on handling these things in your engine? Are you able to replicate, in the prototype, the kinds of movement and collisions you want without relying on Unity-specific constructs? Will you use an existing physics engine? Does anyone on your team have experience doing that? What kind of player input is necessary for a successful game?

The prototype should:

* Be a grey-box prototype
* Let the player to experience your core mechanic
  + Match-3, jetpack flying, throwing objects, etc.
* Allow you (the developers) to determine what your C++ engine will need:
  + Physics?
  + Raycasting?
  + Rigidbodies?
  + Trigger volumes?
  + Collision Detection?
  + Mouse/keyboard control?
* Help you to nail down the gameplay code in a language more forgiving than C++

The prototype should NOT:

* Take weeks and weeks to finish
* Have final art or other assets
* Worry about shaders or graphics programming (your engine will do this!)