Project & Portfolio IV

Game Team Startup Guide

# Stage 1 – Establish Team Communication Pathways

* Communication is the key to a successful team project. Even highly skilled/hard-working teams will become ineffectual if everyone just goes off and does their own thing. Lack of communication and/or organization is the #1 killer of projects.

## Sector A – Contact Your Team

* Reach out to your teammates. Talk to them about what classic video games you enjoy and what your favorite part about programming is. (Ex: Finding Bugs, Math/Physics, Building Gameplay Systems, UI, Graphics, etc.)
* One of your members will be defaulted as the Lead Programmer. If you are wondering what being the Lead Programmer entails, check out the “Game Team Roles” document which should be next to this one. If one of your teammates is completely unreachable notify your instructor immediately. Any students who refuse/fail to communicate may be placed into their own team.
* Conduct yourself in a professional manner, you are only a few months away from applying for jobs and potentially working with total strangers. Now is the time to practice these skills.

# Stage 2 – Organize Your Team

* Defining expectations in your team is critical to the project’s success. Two ways to do this is through the selection of programmer roles(responsibilities) and task tracking.

## Sector A – Select Your Roles

* The only mandatory role in the team is the Lead Programmer (you must have at least one). They are responsible for tracking tasks on the Jira board. Default team roles will be discussed in the Discord welcome message.
* Review the Document “Game Team Roles”. As a team, decide how to best divide the various responsibilities of game development. Finally get together and start filling out the “Team Info” tab in the “Game Team Progression Rubric” which is also in this folder.

Preferences:

Andrew –Graphics

Adam – Lead

Sandy – Gameplay

Ralf –Generalist

James – Gameplay

## Sector B – Create a Jira Board

* This game project will be a series of 4 Sprints, one for each week. (Pre-production, Alpha, Beta, Gold)
* Watch the following tutorial and create a Jira Board. This will be used to figure out what needs to be done, how long it should take, who is doing it, etc. Make sure all your team members are added to the Jira Board.

[](https://www.youtube.com/embed/aP7W7zNTM2I?feature=oembed)

* The Lead Programmer should handle the Jira setup, make sure to also invite all your instructors to the Jira board so they can review your progress and give advice as needed. (Instructor emails are in the Syllabus)
* Jira has many complex Agile workflows available. However, due to time constraints we suggest just using the basics of creating/assigning/completing tasks as shown in the video. It is simple to do and still plenty effective.

# Stage 3 – Pick Your Game Mechanics

* In this course (and the degree in general) the focus is on programming game mechanics and supporting systems. Not design, not storytelling or art. To accommodate this your team will be looking to existing games for gameplay logic vs. designing your own mechanics.

**Mechanics:**

1. Shooting arrows
2. Enemies have health / die
3. Pick up items, music reward
4. Moving platforms
5. Inventory screen
6. Life system
7. Scoring system
8. Trigger walls
9. Trigger based enemies
10. World map

## Sector A – Check Out Some Classic Games

* There is a companion document in this folder called “Game Mechanics-Genre Selection”. It details various styles of classic games and estimates appropriate team sizes for each type of game across one month.
* Take Note: You cannot just carbon copy the art, music, sounds, story, or exact source code (assuming you had it) of a commercial product. Those items all fall under intellectual copyright, and you cannot duplicate them.
* However, what you can do is use free art, music/sound etcetera to create your own interpretation of classic mechanics you and your team are excited about. Even if your game’s functionality is identical, the creative aesthetics need to be highly distinct and unique to this project. (Programmer art is totally ok here!)

## Sector B – Make A Decision

* Once you have settled on the type of game you want to make, start discussing what kind of aesthetics you want to go along with these types of mechanics. This will be important to agree upon for your first Sprint.
* If your team can’t agree, then put it to a vote. If the vote doesn’t help, flip a coin. Don’t spend too long choosing your game. This month will fly by, and you will need all the development time you can get!

Dungeon stuff

# Stage 4 – Get The Ball Rolling

* Once you have all decided upon what type of game you are making, (and what it’s going to look and sound like) it’s time to construct a concrete plan and put it into action. Don’t be afraid to tweak your plan if things come up.

1. Understand ECS / Space Game
2. Swap space game vulkan APIs for OpenGl
3. Design Levels / Art Direction
4. Layout core systems
5. Pseudo for game mechanics (what are our mechanics and how do we implement)

## Sector A – Plan The First Sprint

* Schedule your first official development meeting to discuss, create and assign tasks in Jira for the week one Pre-Production Sprint. Each week there will be a project deliverable, where the team will be asked to evaluate the existing state/progress of the project and submit it for external review.
* Each Sprint will recommend various tasks you will probably want to add to the Jira board. At the end of the day however, what is on the board and how you choose to tackle it will be up to all of you.

## Sector B – Determine Your Meeting Schedule

* In the “Team Info” tab in the “Game Team Progression Rubric”, there is a spot for your team to decide when good times are for the team to meet during the week. Successful teams will generally be chatting daily anyway, but having some structured meetings ensures there is a minimum expectation of communication.
* In Agile development, having a daily standup meeting is a proven method for keeping everyone in the loop.

[](https://www.youtube.com/embed/er9gntPjTJU?feature=oembed)

* Having occasional meetings with your instructor present is not required but is highly encouraged. Particularly when your team is experiencing issues that they are having difficulty sorting out on their own.
* If you are having trouble pinning down a good time to meet (this can be particularly challenging for Online students) Consider using free software like <https://whenisgood.net/> or <https://www.when2meet.com/>

# Stage 5 – Let’s Go!

* At this point you know what kind of game you are making; you know what tasks you are responsible for this sprint. All that’s left to do is dive in and start getting things done. Engineering video games is why you are here, let’s take that dream seriously but also try to enjoy yourself!

## Sector A – Study The Base Code

* Each team will start with a shared repository that not only contains important course documents, but also an Example Space Game created with Gateware, the inifile-cpp library and the FLECS API. You are welcome to use this code as the foundation of your game. (And quickly incorporate one of your Level Renderers from 3DCC)
* The code inside may be a little confusing at first. However, rest assured a large part of the early lecture material will revolve around explaining the various systems used in this code and how they are useful in game development. You don’t absolutely have to use FLECS or even the base code, but it **must** utilize GWindow for OS window management at a minimum. (Ensuring this will be a mostly from scratch project)

## Sector B – Don’t Neglect Pre-Production!

* As programmers we tend to want to just dive into the code and start messing around. (At least I hope so!) Spend the time to gather all the needed visual and audio elements of your game. Be thorough, play or watch the game that has inspired your game and make a note of everything you see or hear.
* The more seriously you take this step, the more rapidly your project will come together in the final weeks as you don’t waste time hunting for assets while neglecting critical bugs and features.