Game Team Roles

Project & Portfolio IV

# Overview

This document will highlight the common roles people might experience on a small game team. The roles in this guide are not absolutes. Depending on team size, individuals may need to wear many hats or pick-up the slack when things fall behind for a project to meet deadlines.

# Lead Programmer

Despite the title, a lead programmer doesn’t always have to be the most skilled programmer. In the industry a lead programmer is typically an experienced developer, part programmer and part manager/organizer. In the context of school, this role is more about accepting responsibility vs. having industry experience.

As the lead programmer your primary responsibility is that the game doesn’t get seriously broken or messy. This means code must always be reviewed by at least one peer before it gets merged. Often that means you, but sometimes another team member should be tapped to approve a pull request. For Example: If the gameplay programmer requests a change in the graphics engine, then the graphics programmer should ideally be the one to approve it:

[](https://www.youtube.com/embed/8fx-EaOUK2E?feature=oembed)

To ensure these types of approvals can happen, you will need to talk to your team and request that they use a branch-then-merge workflow. This means team members will need to branch from and then send pull requests to the main branch. It will be up to you to make sure everyone is aware of, understands and *follows* this workflow.

Your secondary responsibility is to use Jira to setup the weekly Sprints and ensure everyone is updating their tasks correctly throughout the week. When not trying to ensure things are running smoothly, you should use your programming skills to support any tasks that might be falling behind or need extra effort to complete.

Scheduling team meetings and stand-ups round out your key responsibilities. As the main point of contact for the team, any major issues should be discussed with your instructor if they cannot be resolved internally.

# GAMEPLAY Programmer

One of the most critical/challenging roles on the team. The primary responsibility of the gameplay programmer is to implement gameplay mechanics, ideally using the FLECS API and an event system. They will also show the team how to effectively apply game architecture so everyone can integrate their work quickly:

[](https://www.youtube.com/embed/Z-CILn2w9K0?feature=oembed)

The secondary responsibility of the gameplay programmer is to quickly adapt to and learn how to use systems designed by other members of the team. For Example: Most games need text/font renderers so that things like game information can be effectively communicated to the user. As soon as the graphics programmer incorporates a working version of drawable text, you will need to learn it so that they can move on to other tasks.

It is important not to confuse the role of a gameplay programmer with that of a gameplay/level scripter. Scripters utilize pre-existing systems in large projects to ensure each level will work as expected. The gameplay programmer is the one who engineers those reusable systems. (e.g., Doors, Lifts, Turrets, Combat)

Low-Level programming and linear algebra skills are critical to this role. This is also an extremely high responsibility position (no mechanics == no game). It can start out slowly but can be very backloaded as multiple things start coming together during the last two Sprints.

# GRAPHICS Programmer

Having just completed a level renderer in the 3DCC course, students typically understand what to expect with this role. Unlike the gameplay programmer this role is very front loaded. Your primary responsibility is to integrate your data-driven rendering tech from 3DCC into the game project. This can be your own, a teammate’s renderer, an amalgam of all of them or even a brand new one if needed. This must be done as fast as possible (Ideally in the first Sprint) so the rest of the team can see the game rendering in 3D.

The secondary responsibility of the graphics programmer is to add capabilities to the renderer to enable visualizing critical functional things like text, debug lines and UI elements. Beyond that, the graphics programmer’s job is to enhance the visuals of the game with things like better lighting (PBR), shadows, particle systems, animated characters, post-processing, bloom... the possibilities are infinite.

[](https://www.youtube.com/embed/l8hKwGT-bDc?feature=oembed)

It is not an accident that the two classes proceeding before this one focused on graphics. Graphics programming can be a difficult domain to master but it exercises many skills critical to writing fast real-time code. If you wish you had more time to explore your favorite graphics API, this might be the role for you.

Once basic 3D rendering is online, the graphics programmer can be pulled to help with other things. Just be aware the visual quality of the final product will suffer if they are not given time/space to work their magic.

# GENERALIST Programmer

A bit of a catch-all role, the generalist programmer is around to shoulder a variety of different tasks. In a small team this could be pretty much anything. Basically, if it’s on the Jira board and not part of another developer’s specialized domain, there is a high chance it will be assigned to you.

This is your primary responsibility. To be a proactive coding whirlwind, ready to tackle bugs and features no matter how large or small. This is a popular entry level/internship job in the industry for a reason. Basically, it lets a studio throw the kitchen sink of low to medium priority bugs and features at you. If you can improve the code base over time, then they know you are an effective developer and to retain you/let you specialize.

[](https://www.youtube.com/embed/uoAhs6-s_sw?feature=oembed)

Without you the bugs and “minor” features will pile up and the final product will be an unpolished buggy/janky mess no matter how nice the graphics are or how tight the gameplay is. Your secondary responsibility is to objectively prioritize what parts of the game need the most help or feel the worst. Apply yourself there so coding time will be most effective. If you just sit around waiting for your teammates to assign you tasks, the project is likely going to fail.

Don’t underestimate this role. Having to tackle a large variety of issues is extremely challenging!

# Role Distribution

Not every team will be exactly four members. What should a team look like based on its size? Teams smaller than 4 people will need to distribute tasks carefully to not overburden specific members wearing multiple hats. Here are some suggested examples of team compositions:

1. Point Team
   * Lead Programmer
2. Line Team
   * Lead Programmer
   * Generalist Programmer
3. Triangle Team
   * Lead Programmer
   * Gameplay Programmer
   * Graphics Programmer
4. Quad Team
   * Lead Programmer
   * Gameplay Programmer
   * Graphics Programmer
   * Generalist Programmer
5. Five+ Team
   * Lead Programmer
   * Gameplay Programmer
   * Graphics Programmer
   * Generalist Programmer
   * If the team feels under skilled in a specific area, add MORE (not less) of those programmers.

# SECONDARY ROLEs

In game development, school, and just life in general unexpected things come up. If your gameplay programmer’s computer dies or the graphics programmer breaks their arm rock climbing, it’s going to cause some issues in the ability to ship the game on time.

Therefore, it is important for each teammate to select a secondary/backup role. Whether it’s for an emergency or just to help a teammate who is falling behind/struggling, the team needs to have a plan in place to quickly adapt.

# Summary

Hey! What about AI/Physics/Audio/Input/Database/Tool/UI/Engine Programmers!?

There are many other specialty programmers out there in large studios, we picked the above roles as they will be most relevant to this project. At the end of the day everyone must contribute to this project for it to succeed. Don’t ignore something just because it doesn’t fit neatly into your role. The project is a culmination of what you achieve as a team, not what you did as an individual, so keep that in mind.

If you are struggling on how to distribute additional categories of tasks here are some *suggestions*:

Lead Programmer: Engine/Database Tasks

Gameplay Programmer: Physics/AI Tasks

Graphics Programmer: Tool/UI Tasks

Generalist Programmer: Input/Audio Tasks

And Remember: Communication is everyone’s responsibility!