Scratch

Victor Lozoya
Alexander Kurian
Christian Mcgovern
Mark Kircher
Robert Dale

Survival-Horror

Scratch is a two-dimensional Survival-Horror game that employs concepts similar to games like Don’t Starve, Project Zomboid, and Darkwood.

The player will be equipped with basic tools to help in surviving and can use tools to craft other survival tools. Zombies will attempt to track down the player and damage them throughout the game.

The game itself is made in c# using libraries from MonoGame and the MonoGame Content Pipeline. Scratch is designed to be fast and easy to play with little system requirements and simple functionality while highlighting the ability to craft, survive, and destroy to win.

Key Architectural Drivers

1. Our requirements included players and enemy interactions, so we created a system for Animating Sprites.
2. The key feature for this game is Inventory and Crafting so we created a system that handles these systems individually.
3. The requirement of movement through a map via camera/player positioning had us create a system for the map tiling/creation.

Creation of other components will likely be systems as well.

Component Based Software Engineering and Entity-component -system

1. https://gamedevelopment.tutsplus.com/articles/unity-now-youre-thinking-with-components--gamedev-12492

2. https://www.gamedev.net/articles/programming/general-and-gameplay-programming/understanding-component-entity-systems-r3013

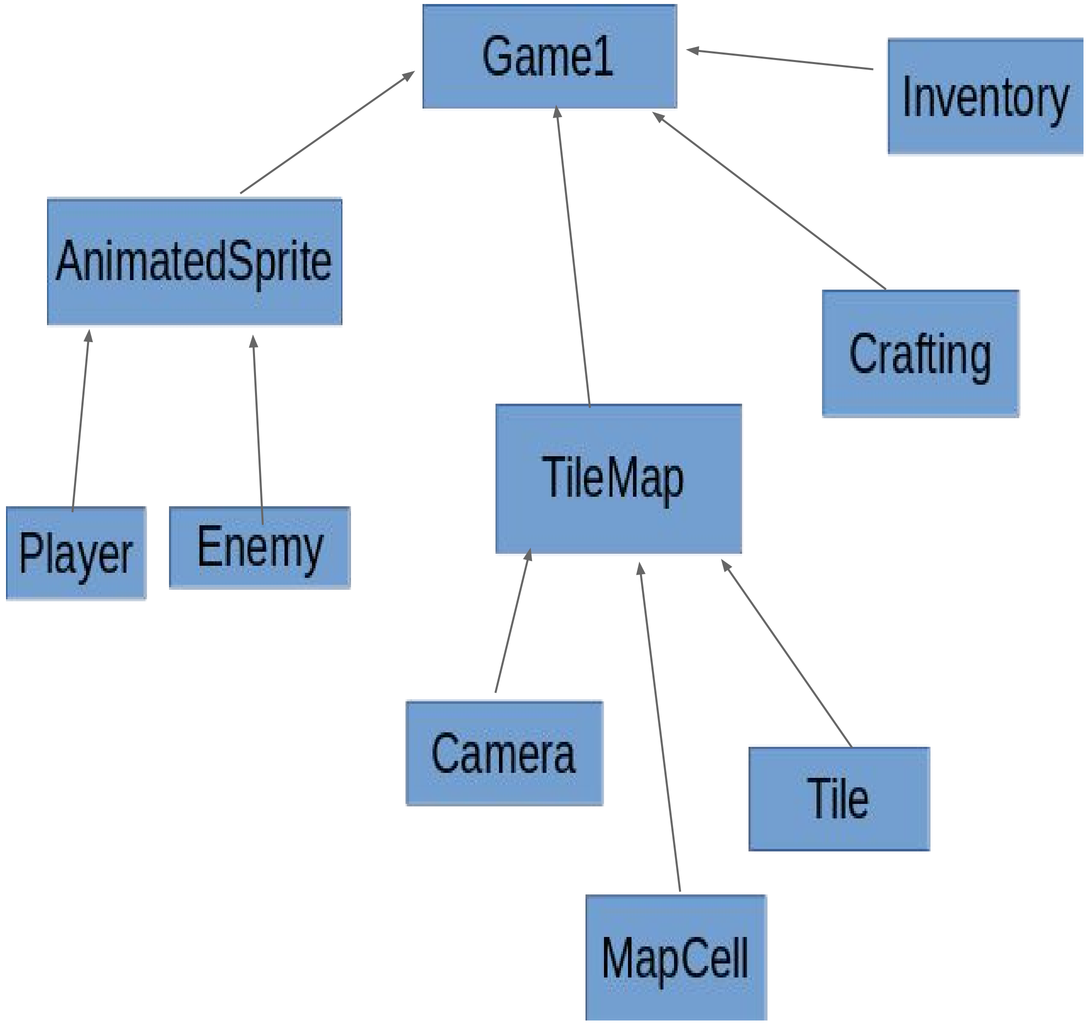
Component Based Software Engineering integrates different but related modules into a single system. A Hotel Reservation system would use CBSE to combine reservations, customer info, credit card processing, and other components together.[1]

Our game uses a similar concept to combine components such as inventory management, player and enemy, projectiles, crafting, and tile mapping into a system that produces the game itself.

The other system we considered was the entity-component-system. We have classes which are the components. The components build entities such as enemies. The game loop in game1 is our overall system that runs on a loop iterating through the components building entities.[2]

- We separated our concerns into individual systems.
- AnimatedSprite takes care of enemy and player interaction.
- TileMap takes care of our map creation and camera movement.
- Inventory helps keep track of what items are collected.
- Crafting takes care of creating items from items.

Architecture Diagram for Scratch - Connecting Many Components



Conclusion

We have identified some issues with Component Based Software Development, but we think it is still the best choice for this project.

Issues/Risks:

- Because our project uses many different systems, there is high subclass coupling.
- Even though our individual systems work by themselves, they may not function properly in a completed project.
- The larger the project becomes, the more complex the system is to maintain.

Questions?