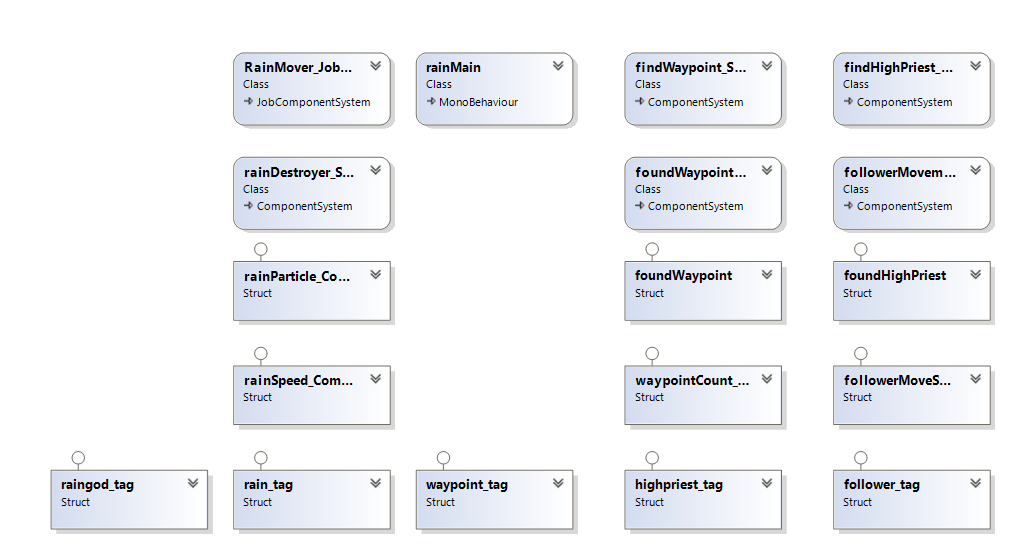
**Final Project [ECS + Jobs]**

**The Rain God**

**Joshua Issac**

**Class Diagram**



**Profiling**

****

**Entities, Components and Systems involved:**

**A] Entities**

1) The Rain God – A snowman that exists for pure fun. One can assume it to be the summoner of the rain.

2) Waypoints – Invisible entities that exist to guide the High Priest (Orange cube) and the followers (Green capsules)

3) The High Priest (Yellow Cube) & The Followers – The High Priest follows the path of sorts created by the waypoints. The Followers follow the High Priest.

4) Land – Provides the base for the rest.

5) Trees – It’s scenic.

6) Rain – Summoned by pressing “Spacebar”. It is the main purpose of this game. On summoning rain, the land eventually gets flooded.

**B] Components**

1) **Tags:** rain\_tag, waypoint\_tag, highpriest\_, follower\_tag , raingod\_tag\

2) foundHighPriest – This component is added to a follower when it “locates”  
the High Priest.

3) foundWaypoint – This component is added to the High Priest when it “locates” the closest Waypoint. It is removed when the found Waypoint is destroyed.

4) rainParticle\_Component – Contains a Particle ID number. (Doesn’t play a role in the game)

5) rainSpeed\_Component – Contains a rainMoveSpeed float that determines the speed of the raindrops that fall.

6) waypointCount\_Component – Waypointcount keeps track of the number of waypoints visited.

7) followerMoveSpeed\_Component – Contains a float that determines the movement speed of the follower.

**C] Systems**

1) RainMover\_JobSystem – C# Jobs system that reduces the Y value of the spawned raindrops in order to create “rainfall”.

2) rainDestroyer\_System – Component System that Destroys raindrops that are at “ground level” – in this case y < .3f

3) findWaypoint\_System – Component System that is used by the High Priest to obtain the location of the closest waypoint.

4) foundWaypointDebug\_System – Component System that is used for Debugging purposes. Draws a line from the HighPriest to the closest waypoint (in the Scene View).

5) PriestMovement\_System – Component System that moves the High Priest to the current closest waypoint. On arrival, the waypoint is destroyed – resulting in the High Priest looking for the next closest waypoint.

6) findHighPriest\_System – Similar to “findWaypoint\_System”, it is a Component System that searches for the High Priest and adds the foundHighPriest component to a follower when it does find the High Priest.

7) followerMovement\_System – Similar to the PriestMovement\_System, it is a Component System that makes followers follow the closest High Priest (in this case, the ONLY High Priest).

**Is the program data-driven and how is it data-driven?**

No, this program is NOT data-driven.

**Did you have to change how you wrote your code to use ECS and the job system?**

**Describe the most difficult thing that you did for your program.**

The part I found most difficult was establishing a path using invisible waypoints, and making sure that the High Priest followed the waypoints, and the Followers followed the High Priest. The System was tricky to create.

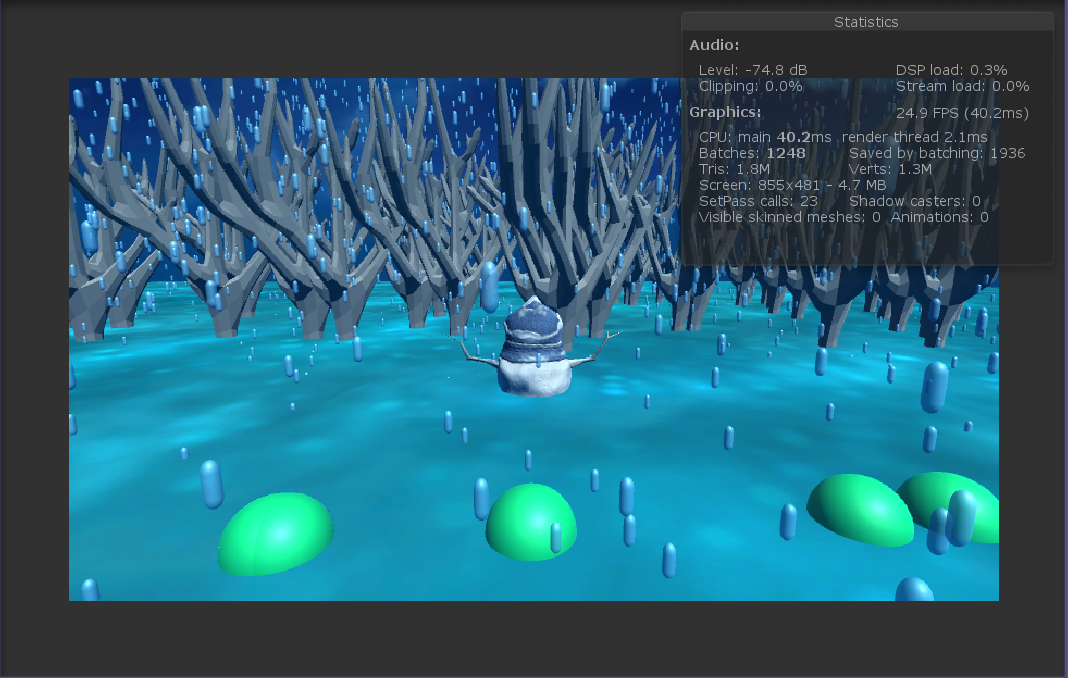
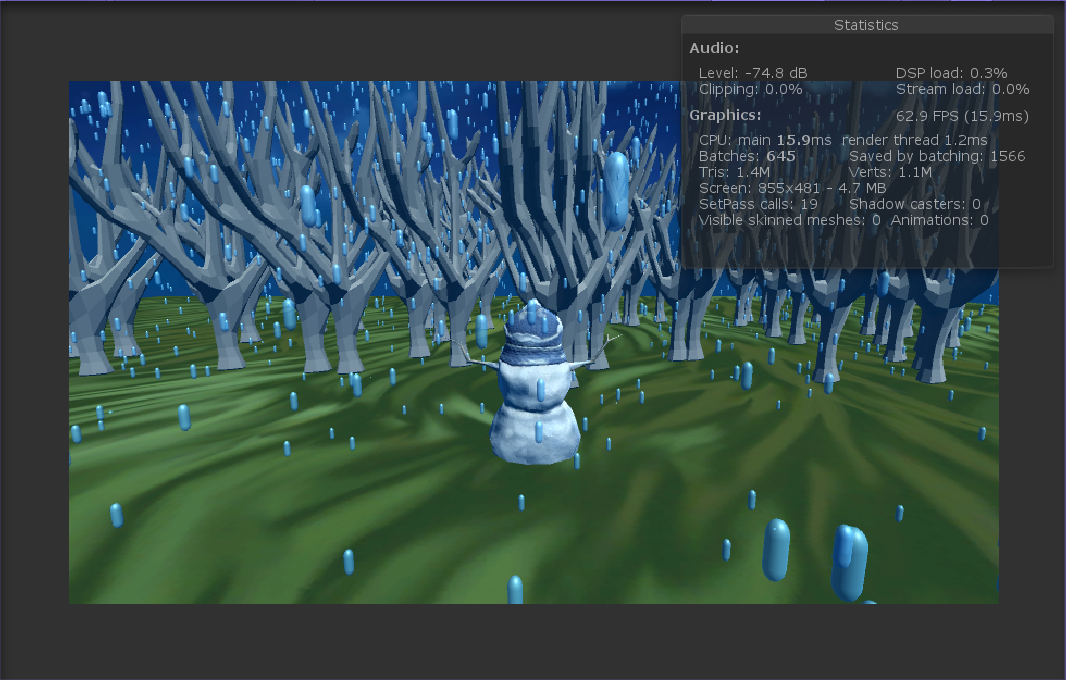
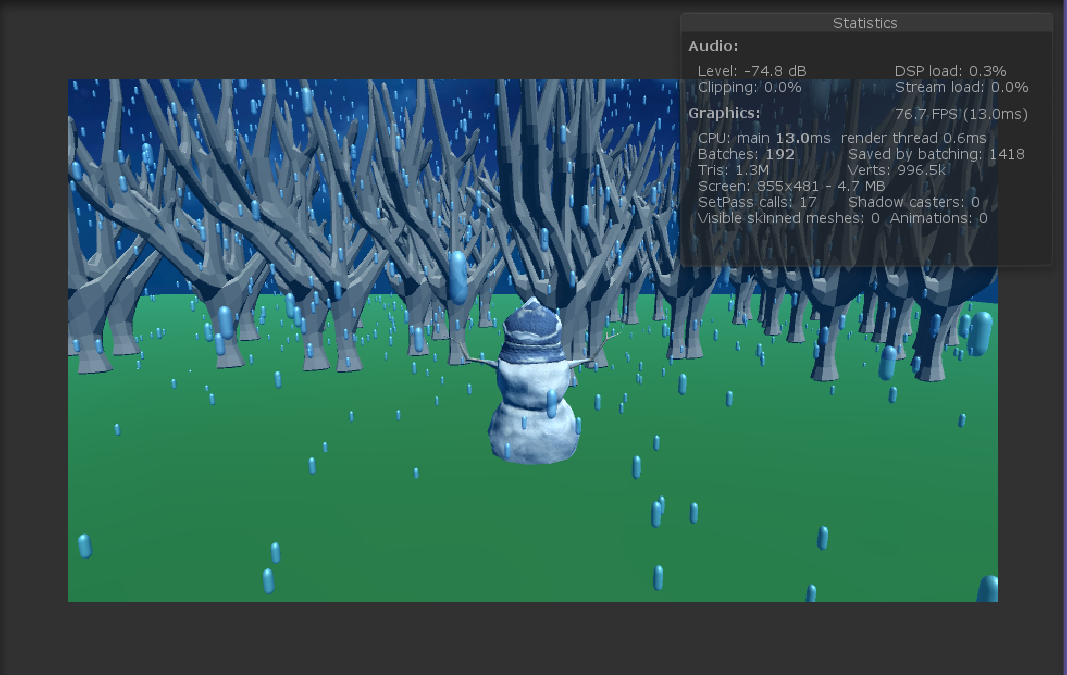
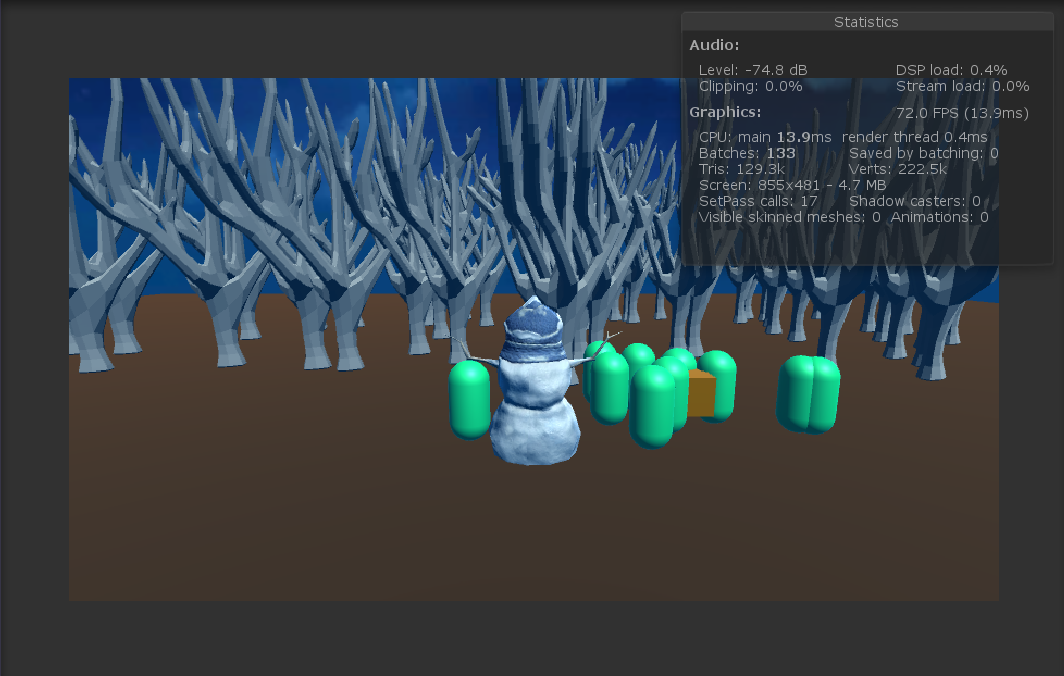
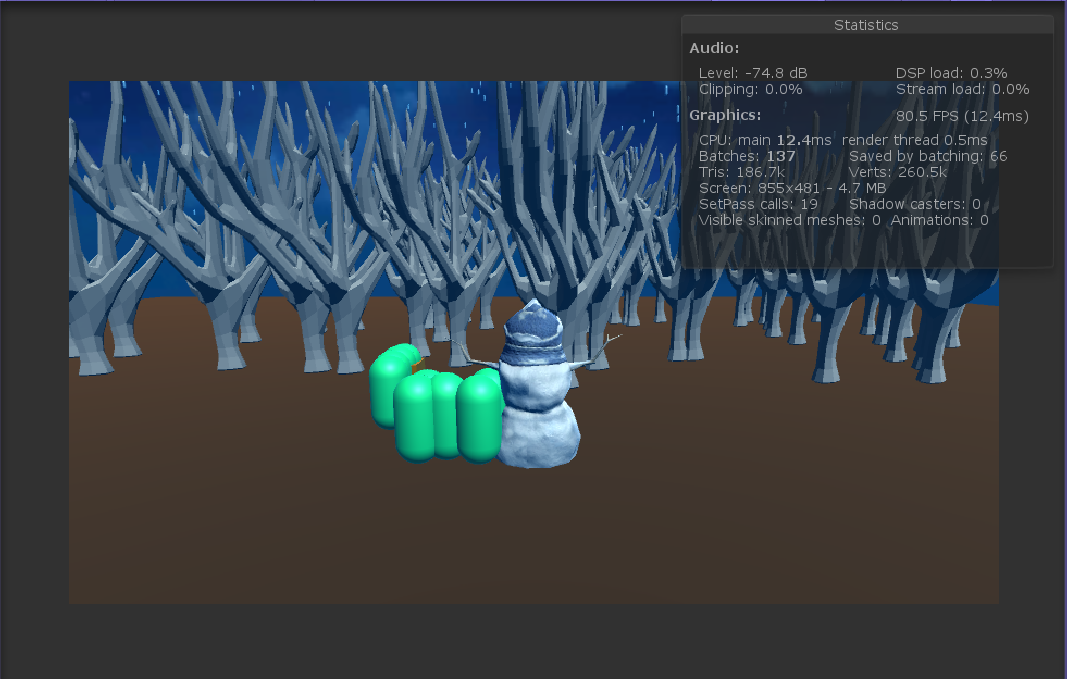
**What parts of Unity ECS did you use and why?**

I used ECS to directly spawn everything as Entities, without the use of a single GameObject. Native arrays were used as containers, and the EntityManager used these containers to spawn various numbers of entities per container.

I used Component Systems to search for and move Entities around as I desired.

I used a Jobs System to create “rainfall”.

**Images**

****