Prototype Crowded game Zombie Horde Shooter

Group 1B
Joel Brieger, Nikos Giakoumoglou, Gianmarco Picarella

Original idea - Tower defense

- Zombie horde attacks the tower from every direction
- The player must protect the tower for as long as possible

What we had last time:

Assumptions we made:

- Low poly models will be cheaper to render.
- No physics colliders are required on the zombies.
- All zombies move towards the same area.
  - No counter-flows
- The player cannot reach the zombies
- Player cannot zoom in

Encountered problems

- Experimental status/Outdated documentation of UMass
- Vertex animations
- UMass avoidance integration with NavMesh
- Large crowds can push agents into obstacles

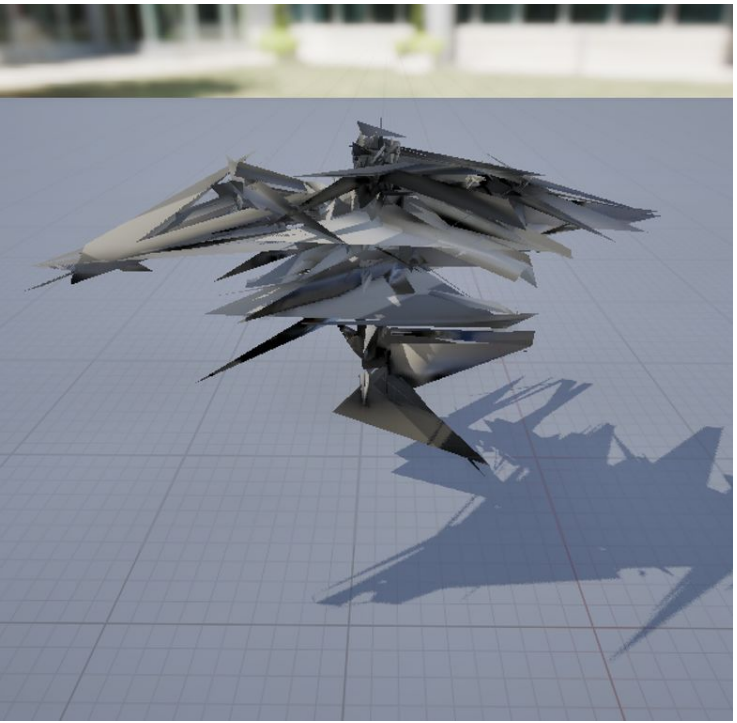
Encountered problems:

Experimental status/Lacking documentation of UMass

- Difficult to parallelize functions
- Difficult to set tick rate of individual processors (Systems in ECS)
- Built in avoidance trait is very inefficient.

Encountered problems:

Vertex Animations



<< How it started

How it went >>

Encountered problems:

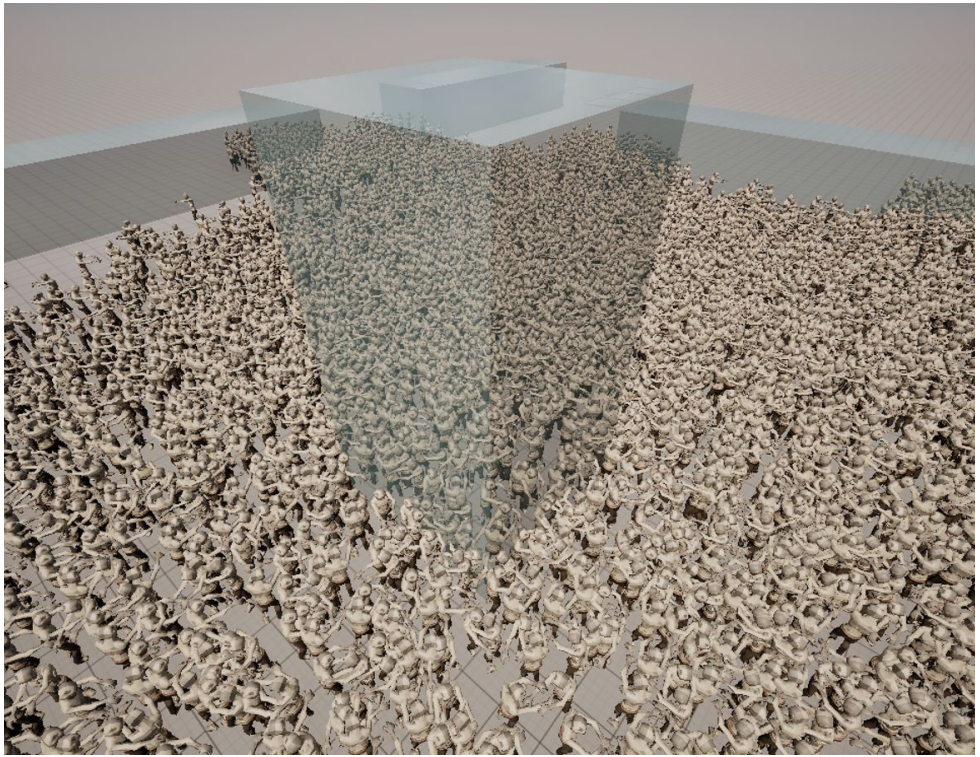
UMass Avoidance Integration With NavMesh (5k agents @50fps)

Encountered problems:

Large Crowds Can Push Agents Into Obstacles

Happens because the entities don’t have collision.

Partially solvable but out of scope.



New Idea - Square defense

- Zombie horde attacks the square from every direction
- The player must protect the square

Making the Game a Game.

Health Status and Zombie Objective

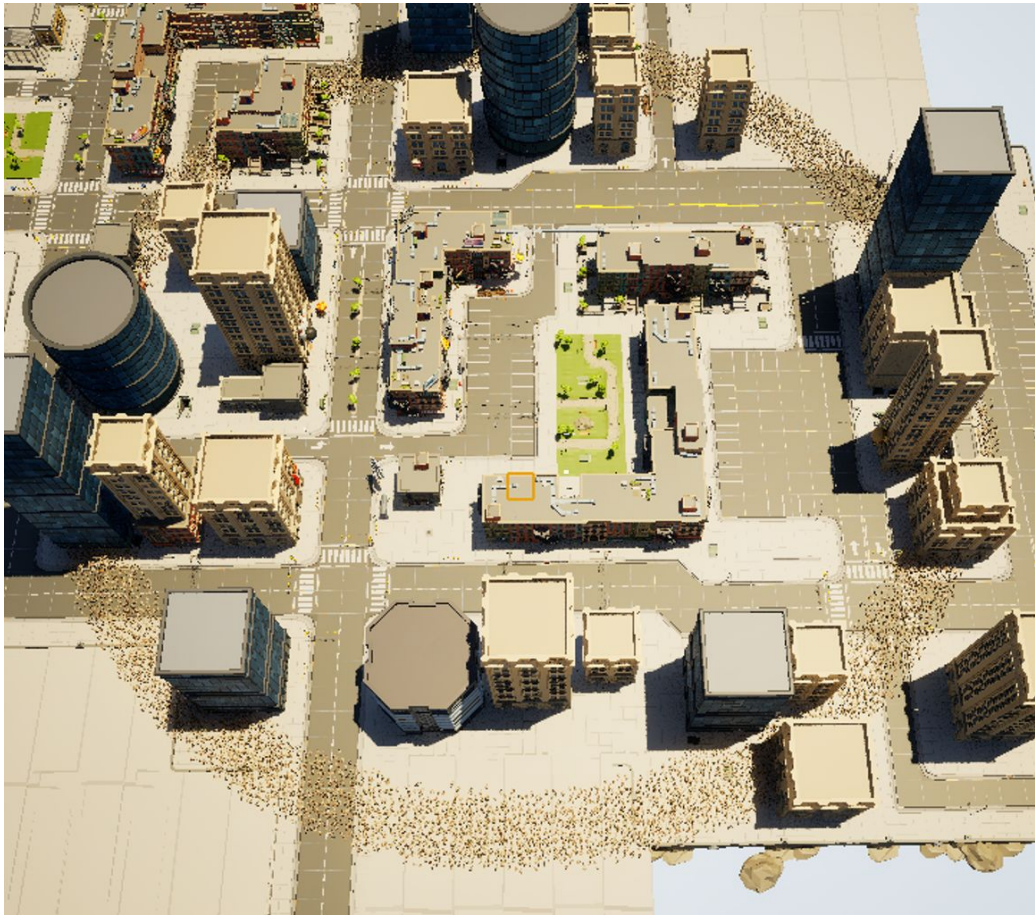
- Start with 100 health.
- Zombies are all trying to navigate to a random position inside the main square
- For every second there are more than 500 zombies inside the main square your health ticks down.

Player Character

- Uses the built-in unreal FPS character controller
- Added super-jump because it is more fun and easy to see the zombies
- Makes for more forgiving gameplay since if you fall off the buildings you can jump back up
- Player has a rocket launcher to fend off the zombies

Zombie Spawn System

- Zombies spawn in a wide ring around the city
- Tests were performed using spawn points but this caused issues with the forces
- Unfortunately we ran out of time to implement a more complex system

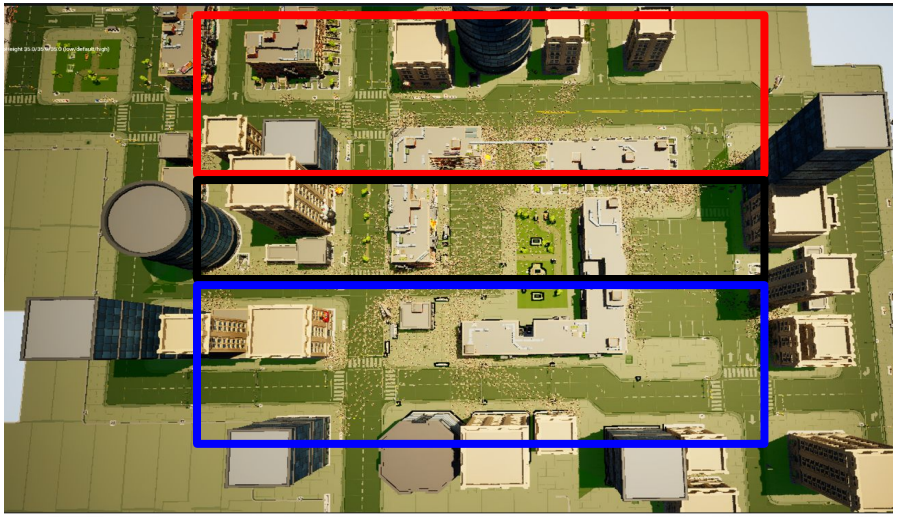


Avoidance

- Initially we used the default avoidance system implemented in mass
- Then we implemented a custom avoidance processor using a faster nearest neighbours algorithm
- We improved the performance of the simulation from 50fps with 5k agents to 30fps with 10k agents
- We used NanoFlann, a KD-tree based nearest neighbours library using a fixed radius search

Additional Optimization

- Each agent shares approximately the same neighbours across subsequent frames
- Agents can be grouped and updated across multiple frames reducing computation time
- Initial tests showed this method allows us to manage more than 40k agents with playable frame rates

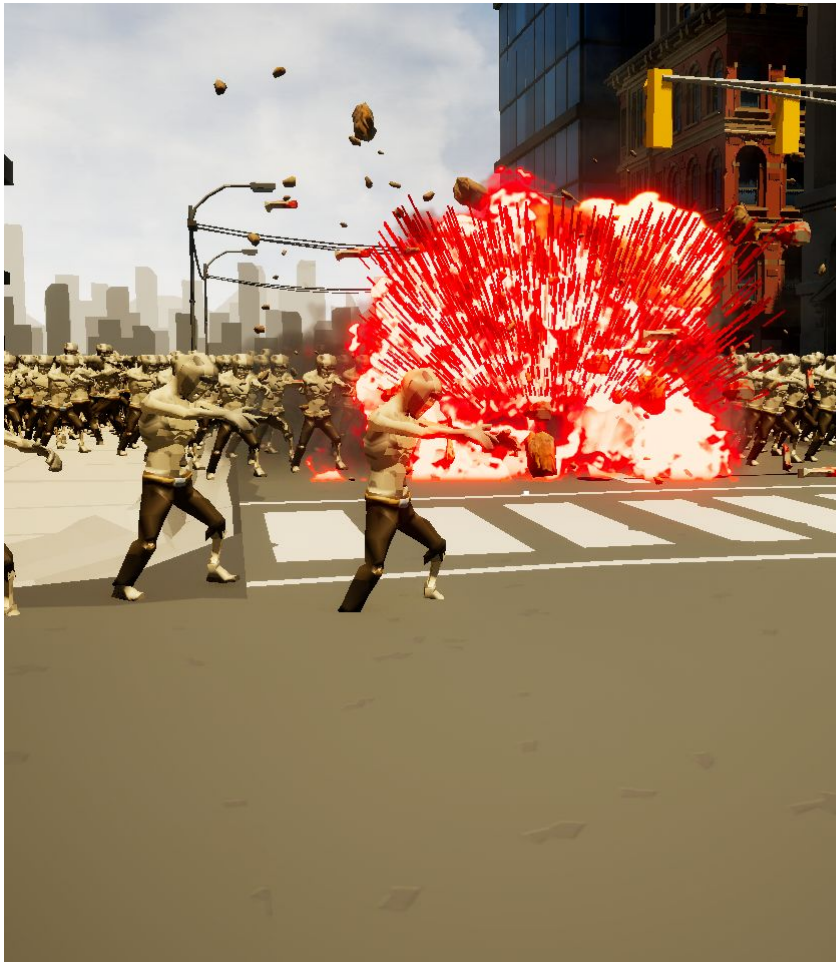


Rocket Launcher Mechanics

- Initially you had to reload the rocket launcher (~2s)
- Then we realised that made the game impossible so we made it full-auto.
- More fun to have a fully automatic rocket launcher
- The projectile hit location and the explosion radius are used to locate the agents that should be deleted using the same functions as NanoFlann

Visual Effects

- Made up of multiple different niagara particle effects
- Rocket launcher explosion effect
- A blood and guts effect that's based on whether or not you hit a zombie
- Initially we tried to implement it so that it would scale the blood and guts based on the number of zombies hit but that was scraped for performance reasons



Limitations, shortcuts, and pitfalls

- Zombies still get stuck on terrain
- Rocket launcher radius is 2-dimensional and drops the z values so you have to hit the ground to kill them.
- The vertex animations cause the zombies to look unimpressive up close.
- Zombies all play the same animation at the same time, with the same texture which looks very monotonic.
- Zombies only play one animation so when they reach their destination they walk on the spot.

Testing Conditions and Results So Far

- Subject to change in the final version
- Tests were performed in the UE5 editor which is not ideal for performance
- Test machine was a RTX 3080 and a Ryzen 5 5600x.
- Tests show we can run 5k agents at ~60fps and 10k at ~30fps

Thank you! Questions?