COMP 3501 Game Project Write-up

Authors: William Clifford and Francis Lavoie

How to play:

The goal of the game is to be the first player to collect 10 chase cubes. The game is played by racing enemy tanks towards the cube's location. Drive over chase cubes to pick them up. The chase cubes can be easily seen from a distance because of the spotlight on them. The player can move using WASD keys and moves the turret with the mouse to aim. Left click will fire a bullet towards the targeted location. Bullets have a very high density and as such have enough mass to knock back enemy tanks. Pressing space-bar will toggle the camera mode between first person and third person. The mouse scroll wheel can be used in third-person view to move away from the tank to see a wider view.

Algorithms, game logic and technical features:

- Quadtree space partitioning: We used this to do frustum culling when chunks of the terrain are not in view, and also to speed up the height lookup function to avoid having to search the entire terrain. It is very slow to build the quad tree at startup, and gets much slower with bigger terrain sizes so we had to settle with smaller terrain size in order to keep load times lower.
- Model loading: We originally used an .OBJ loader written by another student but eventually we wrote our own binary model format based on the struct type we used for our models, which was much faster to load due to just needing to read the binary directly into an array instead of having to parse the data. Our benchmarks showed that this method was roughly 5000 times faster than the .OBJ loader.
- Pitch correction while yawing the tank on a slope: This was to fix the problem that the turret would be hard to control while on slopes. The calculation is a set of cross products and projections to get the angle offset required to correct the pitch to effectively make the yaw occur on the world plane instead of the plane formed by the slope.
- Our turret and tank rotate independently of each other and are hierarchical, the turret following the tank's position as it moves. The tank is moved and turned using the WASD keys and the turret is controlled by the mouse and can pitch and yaw. The first person camera mode places the camera near the turret's position as if you were inside of the turret itself. The third person camera is selected by default and is behind and above the turret.
- Tank hovering is done by applying hooke's law to simulate a spring between the ground and the tank when the tank is below a certain threshold. By using hooke's law the tank will eventually come to rest where the net force caused by gravity and the spring reaches zero.
- There are four types of collisions in our game:
    - Tank vs Tank: Tanks have two spheres defining their collision area, the first is about the turret base and the second is a smaller sphere about the front of the tank. Tank vs tank collision is then done using a sphere-sphere collision model.
    - Tank vs Bullet: Bullets are small enough that we used a point-sphere collision model. Bullets are given enough mass to be able to knockback enemy tanks
    - Tank vs Static objects (rocks): collision between rocks and Tanks is also done using a sphere-sphere collision model with the rocks acting as having infinite mass.
    - Tank vs Chase cube: We used a sphere-sphere collision model, but as cubes are a pick-up item there is no collision resolution
- Our lighting is applied to everything in the world. We chose to use a white global light with specular highlights on the boulders and tanks. We chose a 35% ambient lighting constant to simulate the

light reflected off the terrain.

- The particle effects are spawned from the location of a collision between a bullet and a tank or two tanks. 50 particles are spawned every time a collision is found and at most 1 collision per frame is allowed to spawn a particle effect. There is a hard limit of 10,000 particles in the game to prevent lag. Particles last roughly one second from spawn time and are affected by gravity.
- Each tank is allowed to shoot bullets. AI will shoot a bullet towards the player every 0.25 to 0.75 seconds. The player can shoot bullets using the left mouse button and is not limited in the rate of fire. The player can shoot as fast as they can click. The bullets have a small model and are fired from one of the 16 barrels of the turret. The bullets can collide with enemy tanks to push them back and slow them down momentarily. The bullets move relatively slowly and are not instantaneous and act more like mini-rockets.

List of bugs:

- Collision with static objects is buggy, can get stuck on it or even get into the objects.
- Collision with the side of other tanks is buggy because they are formed with two spheres. Using a capsule collision instead would probably fix that.
- When the game begins, there's a chance that the turret is flipped upside down depending on if the mouse was moved while loading assets. It settles itself in after moving the mouse again though.
- Leaving the bounds of the map has a high chance of glitching the game and seeing a black screen.
- There is a chance that the chase cube can spawn within a static object.
- The camera will shift up and down when going up and down hills because it's linked to the turret's forward direction, which also changes when going up and down hills.
- Pushing an enemy tank up a slope (only happened with AI disabled) will make the enemy tank disappear entirely from the map.
- The bullets are not correctly spawned from the turret barrels due to an incorrect calculation to compensate for the turret rotation.
- The particles are drawn on top of every other world object due to the depth buffer being disabled when drawing them to avoid alpha blend artifacting.

Software design and game design decisions:

- We chose early on to make a game with hover tanks because we wanted to avoid the problem of contact with the ground. This would simplify some of the game calculations and separate our game from some of the others by doing bouncy physics when going up and down slopes and make some of the collision interesting. It also eliminated the need to implement spinning wheels and friction with the ground.
- The chase cube gameplay was inspired by the Rocket Ralley racing section of the game *Rage*. The player chases down beams of light against other AI. Our game has similar gameplay where the focus is on reaching the checkpoints, with the ability to knock away enemies by running into them or shooting at them.
- We chose to use a pre-generated heightmap for the terrain because it would make for easy slopes and is one of the simplest ways to get interesting terrain features. We used a program to generate the terrain for us which made a valley with mountains on the edges which are meant to hide the seams of the height map.
- Much of the framework of our game was based on the Rastertek tutorials, so much of what we have resembles that code. We chose to base ourselves on these tutorials because they gave us a very good framework to build the rest of our game and it implemented many of the graphic engine features we knew we would need in our game.

Distribution of Work:

Francis:

- Graphics
    - Worked on setting up the graphics pipeline along with most of the shaders and D3D features (alpha blending, z-buffer toggles)
    - Text debug and score display
    - Frustum culling
- Models
    - Heavily modified the tank model (which was taken from TurboSquid) to fix some problems with it (flipped normals, disjointed vertices, off-center cockpit) and also split the turret model into it's own model to be able to use them separately
    - Created a bullet model based on the rockets from the turret model
    - Created a cylinder model for the spotlight on the chase cube
    - Added a skysphere model
- Particles
    - Implemented a particle system that spawns on collisions using spherical coordinates for the velocities
- Terrain and Environment
    - Used GeoGen terrain generator to build the height map used in our game
    - Added asteroid models taken from SolCommand as the boulders in the environment
- Input
    - Implemented input handling including key locking to avoid repetition while holding a key when it isn't desired, as well as mouse movement detection to control the camera.
- Game elements
    - Bullet spawning and handling
    - Pitch correction on slopes for the tank
    - Set up the State class for the game objects

William:

- Camera
    - Setup camera to follow the turret's forward vector
    - Allowed scrolling to adjust the amount of distance behind the turret the camera is.
- Movement/Orientation
    - Created a physics model for movement that includes friction
    - Added hooke's law to simulate hovering for the tanks
    - Oriented base of tank to the compensate for changes in the terrain
    - Implemented the preliminary orientation code for the turret
- AI
    - Wrote an algorithm for tanks to move towards the Chase cube
    - Wrote an algorithm for aiming turret at a given target
- Collisions
    - Implemented collision checking for all 4 types of collisions listed in Algorithms and game logic section
    - Implemented collision resolution for tank-tank, tank-bullet, tank-rock collisions
    - For tank-tank collision properly moved tanks to compensate for penetration

Reflection:

Francis:

I'm really glad that I was able to produce something that actually resembles a game this term. My 3501 project last year did not pan out the way I wished it did. We didn't complete the game and it felt very clunky and featureless. I felt that the course didn't properly guide us to make our project. This term was a bit of a redemption for me, I was able to make a game that felt much more complete.

I felt that this game project was the one that I learned the most whilst doing. The 3D graphics world was always very intimidating to me, but it now feels much more approachable having made a 3D game. I'm really satisfied with how our game looks and feels although it is very simplistic.

I hoped to be able to have better particle effects in the game. Since it was put very much to the last minute, they don't look as good as I hoped and they don't currently allow for fading based on time because of weird problems with the vertex buffer. Another of the most disappointing parts is the collisions. They feel very clunky and jagged. Many of the bugs we have are because of time constraints.

The user interface is also very lacking in that it only displays a hacked together mouse cursor (which has no purpose) and debug text in the top left of the screen. Having little distance indicators towards the enemy tanks was an idea. One of the biggest UI features I hoped to have was a minimap that displayed information such as the location of static obstacles, the location of the current checkpoint, the location of the enemies, etc. I would've also liked to have alternate weapons such as a machine gun turret attachment which could be switched to on the fly. It would have been reasonably simple to implement swappable turrets due to how the tank is set up.

I would've liked to have more environment features to make the map more interesting. I wanted to be able to procedurally texture the terrain to give it different looks at different heights and to implement bump mapping to make the terrain look less flat. I wanted to also implement waterbeds in some of the lower areas of the map to make it look less like an arid wasteland. The skybox is currently just a gradient between two colours which looks quite boring. Adding perlin noise clouds was on the wishlist. Changing the lighting over time was also an idea. I wanted to make the sunlight yellower as the sun set and bluer during the night with much lower ambient light. Moving light and placing the sun in the skybox would have been nice as well.

William:

I like the way the hover tank turned out, both in model and in hovering ability. The basic movement of the tank and the gameplay I think are fairly solid (other than the stationary rocks). I'm also very happy with how well the camera and turret movement turned out.

I would have liked to spend more time on collision calculations and implementing any sort of rotational physics. I also would have liked to add other gameplay features: minimap, pickups, and alternative weapons.