Space Race

An asteroids shoot-em-up racer

COMP 2501A Project

Aaron Ramos-Lazette | 101039899
Brian Grickites | 100964255
David Nguyen | 101038464

Description

Space Race is our 2501 racing game, designed around a linear track with shooting mechanics. Obviously with the name of the game, the setting is in the harsh vacuum of space. You, the player control your own personal, customizable Mark I space racer.

As mentioned earlier, the track is of a linear design, the variation we have included on the map is a “map generator” for adding asteroids as obstacles. These obstacles do form a rough track of curves and bends for the player to follow, lest they want to collide violently with an asteroid.

In addition to asteroids, players will also be competing with various other racers with their own unique ships and behaviors. Some are solely looking to race past the player and be the first to finish, while others are more aggressive in nature and will use whatever means necessary to decommission the player from the race.

Various other obstacles (to which only one other has been implemented at the moment) will also affect the player’s race. Namely black holes which can potentially impede the player (or used to slingshot yourself) from making it across.

Each racer is equipped with an individual shooting turret which can dispense a variety of different ammunition. From rockets, gauss cannons to shield projectiles (not all implemented), players and enemies alike can use their turrets to clear themselves a path, destroy their opposition or shield themselves from harm.

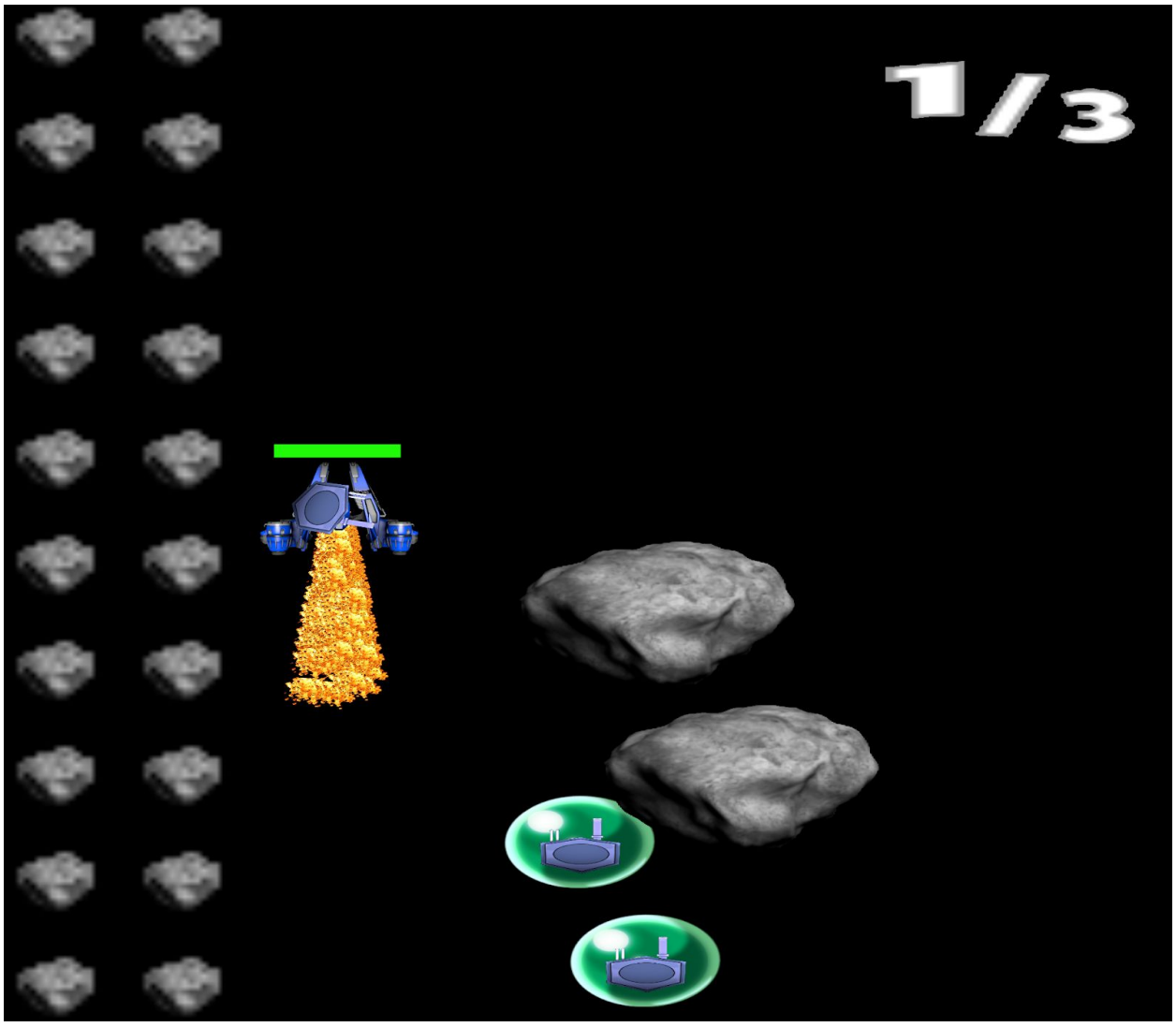
There are various pickups/powerups scattered on the map (only one implemented), some which will boost your speed, provide temporary invulnerability to collisions and such. Other pickups include ammo types which will change your turret ammunition depending on what is picked up.

Each ship can be customized with two varying loadouts in the title screen prior to starting the race (there were planned shop activities to be done outside and in between bouts of racing, currently not implemented).

At the moment the player can choose to vary the mass of their ship, lighter masses allow for faster acceleration but leave you more prone to being affected by an aggressive enemy ramming into you. Heavier masses will allow you to shrug off any

potential aggressor and ram them back with little repercussion at the expense of slower acceleration.

In addition, the player has afterburners located on the sides of their ship for rapid movement to either direction, intended to violently decommission those who wish to overtake you. If you feel particularly vicious, a heavy mass ship with dashing afterburners will wreak havoc on many ships alike. If you're a bit more cunning and sly, maybe the blink teleporters will do you better, instead of dashing you'll instantly teleport a fixed distance to the side.



Player ship with asteroids, enemies with respective turrets on top (note the player ship's turret is rotated to where the mouse cursor is off screen)

How to Play

WASD is the basic control scheme for movement. Due to the vacuum like environment of space, applying a momentum/force in either direction will not be affected by friction. As such movement will be floaty and you'll need to compensate to avoid obstacles. W and S will apply momentum forwards and backwards respectively, and the same goes for A and D but for left and right.

Q and E keys are your bashing/dashing keys, which applies a much stronger force left and right respectively. This is intended as a tool to either bash your competitors off the road, or avoid them entirely (in the case of blinking rather than dashing).

Your turret and rotate according to the position of your mouse on screen. Pressing right or left click will fire the respective projectile to where the crosshair was located.

Avoid asteroids and other competitors with a combination of maneuvers and shooting skills to be the first to reach the finish line!

Requirements

Technical

- OpenGL graphics, C++
  Written in Visual Studio 2017, C++ with OpenGL

- Physics-based movement, collisions
  Ships (and asteroids) move via momentum and have their momentum affected appropriately during collisions.

- Movement through transformations
  Turret rotates independently of the ship (while still translating with the ship's movement)

- Particle system effect
  Flame trail from player ship for particles

- Finite state machine for AI; multiple game states (e.g. store)
  Titlescreen/store is present at the beginning of the game prior to starting the race
  Game Ai can easily switch between either aggressive or pacifist. Pacifist racer has two states of active avoidance and “goal orientation”. I.e. once they pass the player their only concern is staying ahead.

- Multiple game entities with different behaviors
  Two different enemy ships with differing behaviors
  Asteroids have their own static stationary behavior
  Black holes as terrain entities can impede the players obstacle avoidance or aid them in speed

- Multiple terrain types with different effects
  Outer bound of asteroid field actively impedes momentum and movement of ships
  Black holes as mentioned above can be used as a slingshot for speed or actively impede your maneuvers

- Gameworld larger than screen
  Self-explanatory, gameworld is larger than screen, camera is oriented around the player as they move across the racetrack

Gameplay

- Racing game - first to the finish line wins
  Dying will result in a game over, being the first to finish is the only way to win

- Multiple racetracks
  Racetracks can be changed from the codebase by changing the Map.txt file the program loads from. The track as mentioned earlier varies by asteroid placement to form a rough path to follow

- Power ups of some sort
  Speed power up/pickup the only power up currently implemented, provides a boost of speed to the player when the ship collides with it

- HUD showing status, resources
  While no financial resources have been implemented (ammo was also planned

but time constraints and bugs prevented us from getting it fully functional), there is a health bar which indicates the players health (which is affected from going out of bounds and in the future should be affected by collisions and being shot at).

- Nontrivial decisionmaking: action game, but not only action
  From the title screen, the player can choose various masses for their ship and their bashing style. This allows a player to customize their playstyle to what they feel is best for their mindset.

Features that go above and beyond

Our map generation is infinitely scalable and customizable. We take particular pride in this accomplishment as it allows for developers (and to a degree modders) to easily create custom maps of varying difficulty for the AI and players.

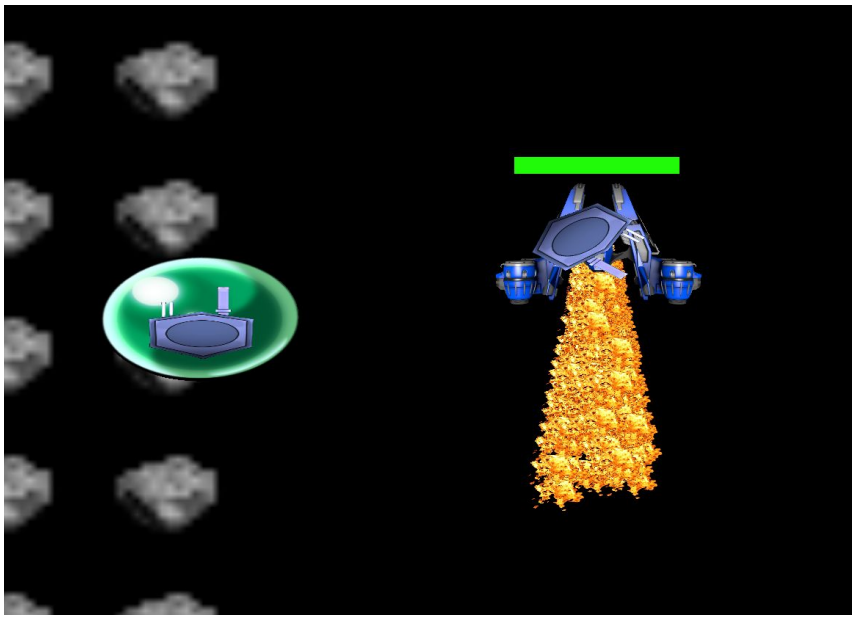
Design Notes

Spaceships are the vessels which players and the AI use to compete in the race of the game. Their behavior can either be predetermined or player controlled (in the case of a player ship). Ships can collide with other solid entities that exist in the track and will rebound accordingly with respect to the mass of the colliding objects. In the case of asteroids which have an arbitrarily large mass, they aren't displaced by the ship to asteroid collision.

Movement of the ship is done via a momentum applied in four cardinal directions. For design constraints, the momentum moving back is capped as there is no logical reason to backtrack on a linear path.

In addition as mentioned above, ships are capable of applying a rapid momentum in either the left or right direction to bash their opponents (either off the track or into obstacles). We designed this feature in mind for allowing some more creativity in

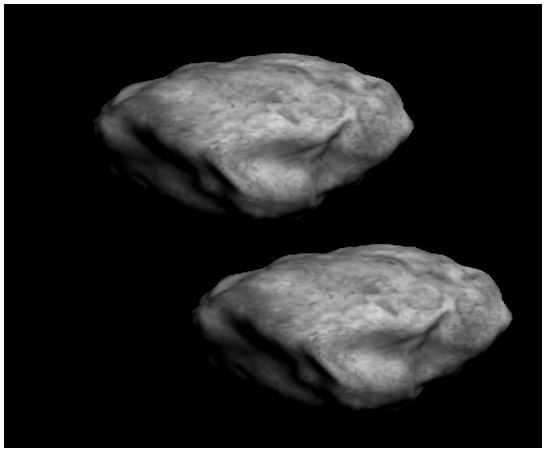
gameplay rather than just simply scraping against competitor ships. In addition it provides some potential for either aggressive and defensive gameplay.



Enemy ship left, player ship right

Asteroids are the main obstacle in the game and in addition form both soft and hard boundaries of the racetrack. While at the moment their only behavior is static and stationary, there were plans to have asteroids that could be moving across the track as dynamic hazards to ships.

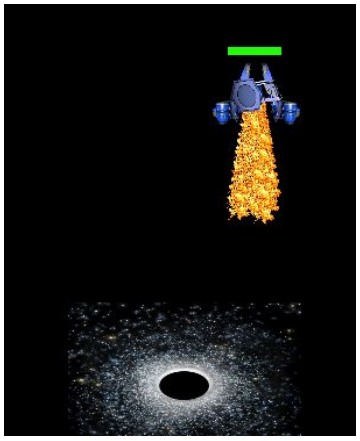
The design decision behind making asteroids both a track definer and an obstacle was to allow tracks to have a very rough outline to follow. The rough outline would however not completely restrict players and AI alike with the means of maneuvering past the asteroids. To simplify collisions with asteroids to ship, ships are displaced from the asteroids, but asteroids are not due to an arbitrarily large mass.



A pair of asteroids

Black holes are one of the secondary obstacles that exist on the track. Their purpose is to add variation in obstacles ships will have to circumvent (or exploit) to win the race. The black hole applies a momentum towards its origin (like an actual black hole with its own gravity well). Players and competitor ships alike will find their navigation compromised, while the more adept can potentially use the gravity well as a slingshot for an extra kick of speed.

As mentioned in the previous paragraph, the black hole was created to add more variation in map obstacles. However unlike asteroids which have a purely “negative” presence to ships, black holes act as a double edged sword provided an adept player/AI can exploit the behavior behind them.



Black hole to the bottom affecting the player’s escape momentum

Overall design decisions -

The idea of basing the game setting in space with asteroids was to minimize work done by being able to recycle earlier code related to the asteroids assignment 1 for 2501. As mentioned before with regards to asteroids, we wanted to create rough outlines for ships to follow, but not explicit concrete routes so that variation in player and AI mindset could show through.

The shooting mechanic (albeit dysfunctional at the moment) was decided to allow more variation in gameplay, similar to how Mario Kart handles pick ups and items, we wanted the turrets on ships to better reflect a specific ship's playstyle. In addition, they could use their turrets to carve their own path into the track rather than rely on the rough path already predefined for them.

While originally, there were plans to allow rotation of the ship and a more curvy/bendy track, we agreed on creating a linear track with less extreme curves to simplify gameplay and to ease player movement. Simplifying gameplay helped us minimize the work needed to be done, and ease of player movement was a consideration as pressing the Q and E keys would bash left and right respectively, but that's only if you were looking upwards. Factor in if the player were looking either left, right or even downwards, and Q and E bash keys could leave the player disorientated and send them off the map (when they didn't want to).

Implementation notes

One of the algorithms we used was directly related from the Impulse collision detection/resolution lecture. More specifically with regards to momentum and adjusting the momentum of the two colliding objects based on their velocity and the inverse of their mass.

Known Bugs & Limitations

Despite ambitious plans for shooting mechanics, refactoring movement to be physics/collision based had utterly broke our previous shooting features. As such all

previously mentioned features regarding shooting have unfortunately been vetoed until a later build.

With regards to black holes, due to backwards momentum being effectively 0 (as it did not make sense to move backwards on a linear track), you cannot be sucked back towards a black hole. In other words once your Y coordinate has exceeded the black hole's Y coordinates, the force it imparts on ships is nil.

Assets

Asteroid:

Author: phaelax
https://opengameart.org/content/asteroids

Black Hole:

Photo Credit: M. Weiss
https://www.scientificamerican.com/article/midsize-black-hole-found-hiding-in-globular-cluster1/

Numbers/text:

Author: Arsonide
https://opengameart.org/content/platformer-art-replacement-gui-text

Flame Particle Jet:

Author: Keith333
https://opengameart.org/content/flame-particle-set-4-in-total

Turret:

Author: mauriku
https://opengameart.org/content/turn-based-strategy-ground-units-set-sand

*Ship, Saw, and Enemy textures provided from base project

Comments on Individual Contributions

***David*:** Personally I feel as though I did my part sufficiently. I had done the tasks I had volunteered for and the tasks I personally selected. I compensated for my lack of work as soon as possible with regards to coding and writing documentation for the group. Aaron and Brian worked excellently with myself and each other, and I have no complaints to their work ethics and efforts contributed to the project even in the face of life priorities.

If we wanted to go into detail, Aaron's work on arranging the class hierarchy for the project was a god send for us (although it proved to be a double edged sword as soon as we realized how complex it got). In addition he provided numerous assets for us to use off of several websites and got major functionality such as maps and shooting (before we broke it).

As for Brian, similarly his work on the AI and power ups was necessary and awesome. It added a sense of depth and variation to our gameplay beyond a simple linear racing track game. Despite life priorities interrupting his work ethic part way through the final crunch, he added much needed features to the game and did his part.

Aaron: Overall, I was impressed with our group. This was a busy semester for each of us for school, work, personal life, and interviews. I feel we were understanding of each other's schedules and hustled when necessary. Our group worked together well, and between the three of us, we recovered from the majority of our missteps.

I looked after a lot of the development outline and some of the task distribution. It helped to ensure that we stayed on task or made up time when necessary. My goal throughout the development process was to ensure that we were as organized as possible. I wanted to ensure the smooth integration of everyone's work into the larger project. To enable a streamlined process, I made sure we were all comfortable with Git source control.

David was fundamental in the process -- he was eager to take on tasks and had quick turnaround time to deliver working features. He built the majority of the physics systems and was easily able to adjust the dynamics to better suit the long-term project. As well, he provided several of the aesthetic flairs that helped bring the project together.

Brian built the outline of the enemy AI systems that, after minor adjustments, invited more dynamic gameplay than I had expected. As well, the power ups offer another layer of gameplay we hoped to achieve. I'd love if we could've given more focus to them, but happy they were implemented.

Brian: The group members I had for this project were fantastic. Despite the very busy semester, the summer job hunt, and other life commitments David and Aaron did exceptional work in the tasks which they volunteered for.

Aaron, in arranging the class hierarchy, set the foundation for much of the work which I and David were able to do. Beyond that he found art assets for us to use, and generally managed to keep us on track. He also streamlined our work flow through ensuring we were comfortable with Git.

David's work on the physics systems similarly made our collective lives much easier, making a system that was easy to understand, use, and add to. He also took it upon himself to compose much of the documentation for the game, which was greatly appreciated by all.

As for the parts which I contributed to, I wish I could have done more. While the AI systems were functional they were more limited than what I had initially intended, and the powerups, again, while functional were limited in their scope. I wish I could have give more attention to them but am reasonably happy with what was achieved.

Project Post-mortem

David: As for what I liked and didn't like about the project, I was personally happy to be with the group I was placed with as we were all extremely reasonable with expectations and had strong work ethics. I enjoyed the brainstorming and idea forming perspective for the project, especially with our decisions regarding an asteroid shoot em up racer game. As for parts I did not particularly like about the project, it was more the fact this whole semester our understanding of OpenGL was extremely subpar. Which did not help as the majority of our project involved a lot of self learning of OpenGL which had crippled our productivity.

As for things I wanted to do differently, I felt as though we could've worked either for longer and more often, but we were often burdened with workloads from other

classes. As much as I can say that in hindsight, I personally believe that our work ethic (which consisted of 1-2 day 12hr+ programming sessions every 1-2 weeks) was sufficient for hitting above the bare minimum, perhaps not the most effective. We had schedules we would've liked to work off, but life and various other school commitments prevented us from following said schedule. As such, I personally would've liked if we managed to do more work on the project, but I would've rathered that the workload from our other courses (including 2501 assignments) was greatly lessened.

As for particular problems, there were none that came up aside from life getting in the way for some of us, but that's expected from any long term project. Refactoring some older functionality created at the beginning of the project broke some key features, so maybe better communication at what was being refactored may have helped (although it is hard to say in hindsight we knew what would and wouldn't break). Concludingly, I feel as though there was not much we could've done differently aside from a bit more work, and maybe next time a few less assignments and life priorities getting in our way.

Aaron: I'll second most of David's thoughts regarding my happiness with the group, and our design decisions. Furthermore, I will reiterate his frustrations with OpenGL. I feel that learning the various features over the course of semester to integrate into our project worked in theory, but fell short in practice. A great deal of our issues were born from a lack of comprehension of the base code required to get the graphics library on its feet. The second, smaller issue (for myself) was due to learning C++ this semester alongside the project.

I felt that our communal cram sessions were effective, especially to ensure that we were all working on the same project and integrating the most important features consistently. They were, however, draining and were perhaps to similar to crunch times to produce the most straightforward code possible. However, I feel they were necessary to keep up with our development schedule. I felt that the difficulty and time involvement of the course assignments very much got in the way of our productivity on the project. While they may not have been intended to be the most involved, there were a great deal of complications between our only recent understanding of the language and libraries, coupled with the subpar documentation for specific cases in OpenGL. What documentation did exist for OpenGL tended to be unclear or differ quite strongly from (and be in conflict with) the base code we were working from.

Once again, I'll second David's sentiments about particular problems. He successfully sums up several of our troubles.

Brian: I was very pleased with my group as the others have mentioned, all performed exceptionally within their roles, and had strong work ethics. Their frustrations I also shared: learning OpenGL turned into much more of a hurdle than was expected at the beginning of the year, and the continued distractions (whether other classes or course assignments) further distracted from the game.

The crunch sessions, while productive, seemed to burn some of us out by the end of them. More consistent work sessions with shorter periods would probably be preferable for future projects. In the future we should also probably attempt to create more realistic schedules, taking into account the fact that life and other school commitments are bound to crop up. There were also several issues that reared their heads in regards to learning OpenGL from scratch, such as knowing what we wanted to do conceptually, and in some cases mathematically, but not knowing how to implement it technically, in some cases on account of the arcane documentation available for OpenGL.

The sentiments expressed about specific problems cover my viewpoint on the troubles we faced, so I don't have much to add there.