Ernest 'iFire' Lee's Journal - Chibifire

K. S. Ernest (iFire) Lee

2017-04-17

Contents

Chapter 1

Introduction

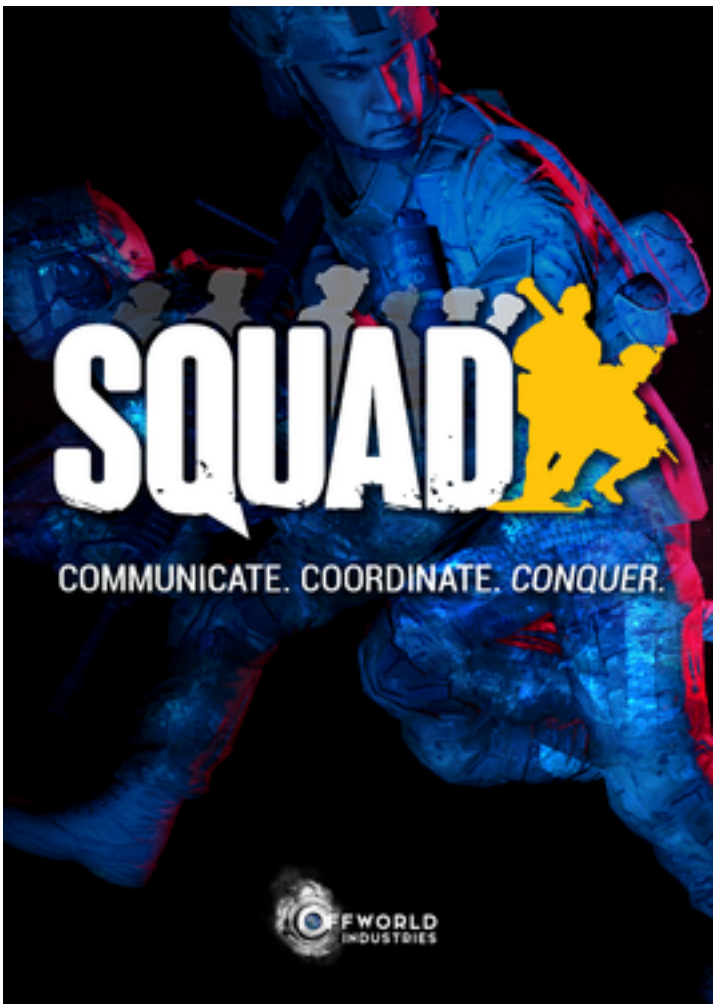
This is chibifire.com where I post my research notes.

Chapter 2

Portfolio

2.1 Current Projects:

Squad - Offworld Industries:



Programmer

2.2 Previous Projects:

2.2.1 Heavy Gear Assault - Mektek Studos / Stompybot:

Systems Professional

2.2.2 Jam at iamagamer:

Penny Farthing by Team Penny

For the `@iamagamer` game jam, it's a game about a little girl full of mirth and ready to get her meadow frolic on.

Credits

Michelle Clough Michelle Buch Nick Irvine Shih Oon Liong Ernest Lee

http://jam.iamagamer.ca/submissions/80-penny-farthing

Chapter 3

Physics in Super Jet Racing - 29 June 2015

This is a guest article from ReactorScram.

I'm ReactorScram, I've been doing Ludum Dare since April 2013. This game is a followup to my LD27 entry, "Jet Racing". People complained it was too 2D so I made 3D physics for this "Super Jet Racing".

In my game, you race a hovercar. Originally I wanted to make a game like F-Zero where the car sticks to the track and is able to hang upside-down. I could still do this, but it seemed making a big F-Zero track would be time-consuming and hard to test.

The level is only about half-done, but I imagine it existing inside a stadium, like they race monster trucks in. There's concrete barriers marking the edges of the track, and right before the start / finish line is a one-kilometer wall, angled at 85 degrees, with a checkpoint at the top. The car has a high thrust-weight ratio, and the player is expected to drive straight up the wall, pass the checkpoint, and fall back down.

As my physics method in my custom engine, I used discrete collision detection and explicit Euler integration. Discrete collision detection means I was too lazy to do continuous collision.

Every frame I check whether the car is inside a triangle, then I push it out. I do this up to 20 times, in case the car is in multiple triangles.

Apparently, you must use an epsilon for this, otherwise the car may keep pushing itself out of a triangle that it's not really in. That one triangle would eat up all the collision resolve steps and leave nothing for the other triangles. During testing, this manifested as: running into a wall and the car would fall through the ground.

When I push the car out of a triangle, I also zero its velocity along the normal of the triangle to simulate an inelastic collision.

There wasn't any issues with penetration other than falling through the ground.

Near the beginning, I had implemented the AABB check before the triangle check, so briefly the car was not colliding with inclined planes properly.

I didn't encounter problems with the car rocketing into the air. Even though I'm using explicit Euler, the simulation is very simple and it doesn't seem to explode. I'm also doing physics at 120 FPS so tunneling has not been an issue. Once I saw it seem to teleport, but it hasn't happened since then. I don't know what it was.

It helps that the track doesn't have any sharp edges where the car might get caught. There's also no other moving objects than the car, although I'd like to add enemy cars, ghost cars or multiplayer someday.

My method is notable because it was hand-written in Lua over the course of a few days.

Cowritten by Ernest 'iFire' Lee

3.1 Additional Notes

Recently Devlin noticed the car's turning looks too "tight", that it does not drift enough. I had already allowed the car to drift in theory.

Every frame, car's sideways velocity is measured and friction is applied to it. If the velocity is small, it will be set to zero, otherwise it will decrease linearly from the friction. The friction was changed so that the threshold for setting the sideways velocity to zero is higher than the amount by which the velocity linearly decreases.

This should simulate a car's tires having high static friction but low kinetic friction. (Even though it's a hovercraft). During a normal turn, the car will grip the road, but if the centrifugal force of the turn becomes too high, the grip will suddenly break and the car will drift. There needs to have some different kinds of curves in the track to test whether this feels right.

For the car to drive up walls, its rotation needs to be changed so the Z vector of the car matches the normal of the wall. As part of the collision detection, a list of normals is created of any walls or floors (the engine doesn't differentiate them) that the car is touching. The game chooses the normal closest to the car's previous Z vector and assumes that the car should drive along that surface. Afterwards, the game interpolates the car's quaternion so that the car will gradually rotate to drive along the wall / floor.

As a result of this, the car pitches back when it drives off of a ramp. If the car jumps off a ramp and turn 90 degrees left or right, the car will be turning within the plane of the ramp, and the camera will show the world tilted slightly sideways as the car drifts through the air.

The car is modelled as a sphere of diameter 5 meters. Maybe this is the right ballpark for a racing hovercar, because Formula One cars are 5 meters long. To make sure that the car wouldn't tunnel through anything, I went to Wikipedia and measured the speed / body length ratios of Formula One cars and SR-71s.

I don't remember the exact numbers, but in 1 / 120th of a second, an F1 car only moves roughly 80 cm. An SR-71 moves a number of meters, but SR-71s are also 107 feet long.

In Super Jet Racing, the car peaks around 3 meters per frame, which I think is 1200 scale kilometers per hour. I might need to change the graphics to make it feel faster.

3.1.1 Grid System

To accomodate large tracks, I use a 3D grid. When the track is loaded, the game iterates over every triangle, placing it into grid buckets based on its AABB. When the game is running, it uses the same function to place the car into grid buckets. Then for collision detection, it only considers triangles within the car's grid buckets.

The current track has about 1,400 triangles so it is useless to check triangles that are nowhere near the player.

A benchmark was setup that spawns the car, then idles for 100,000 physics frames.

Grid on: - 18.7 seconds - less than 0.2 milliseconds per physics frame, including the time needed to set up the grid.

Grid off: - 2 minutes and 43 seconds: - 1.63 milliseconds per frame.

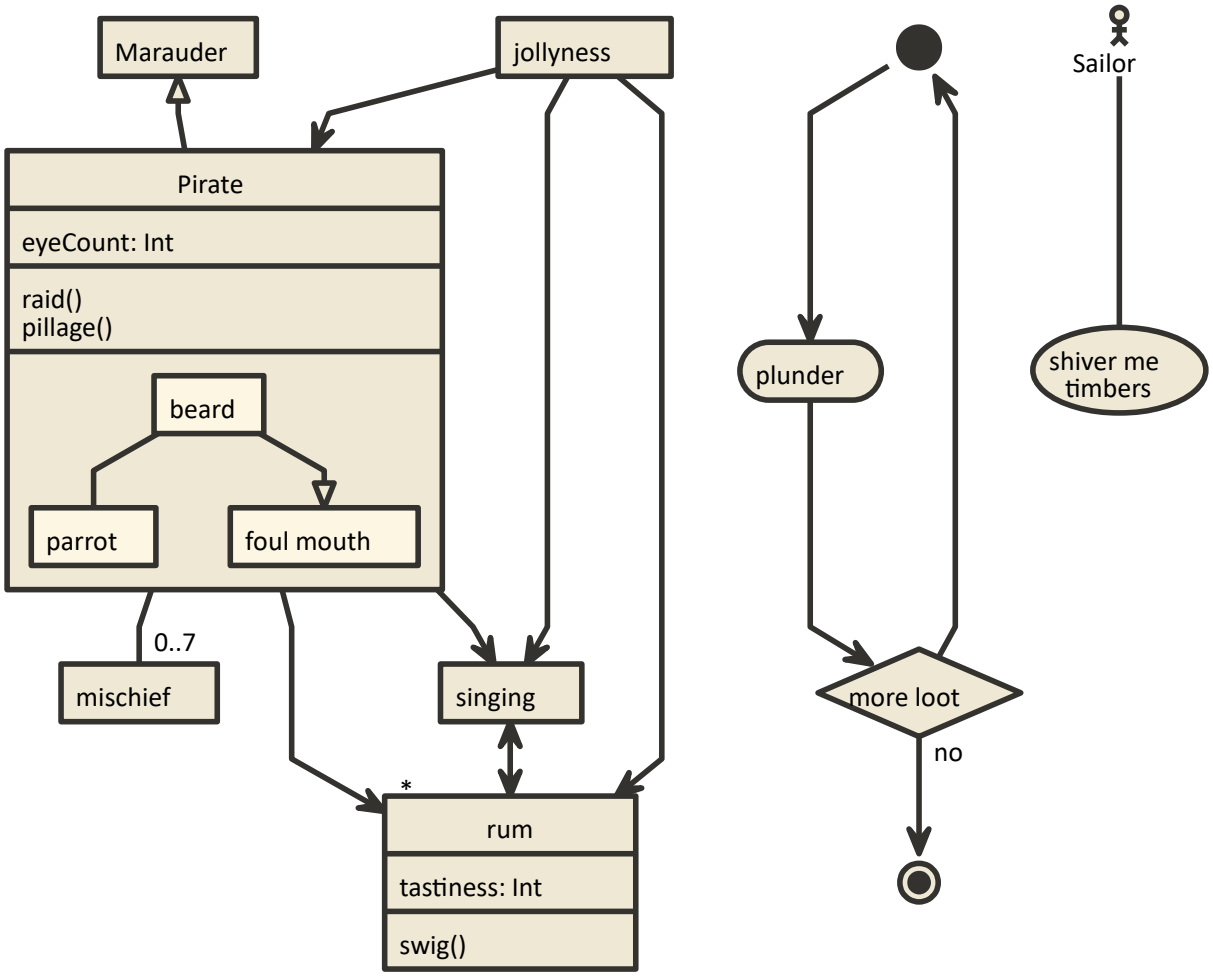
Right now the game is not hurting for milliseconds, but on a weaker CPU, or on a bigger track, or with multiple players, the grid will be a necessity.

Chapter 4

Pirate Nomnoml

Sunday, April 16, 2017 4:11:05 PM

Nomnoml makes uml diagrams.



Bibliography