Rachel Lowe; student no. 2100816

Rl18730@essex.ac.uk

Abstract

In which I discuss ‘CRAPPY’ (the Cool Realism-Adjacent Physics Package, Y’know?), along with ‘A Scientific Interpretation of Daily Life in the Space Towing Industry circa 3052 CE’, a game which I produced using CRAPPY.

A report on CRAPPY (the Cool Realism-Adjacent Physics Package, Y’know?) and the game I produced with it.

Or ‘Misadventures in Misapplied Mathematics’

# Game Description

*A Scientific Interpretation of Daily Life in the Space Towing Industry Circa 3052 CE* (or, for the sake of everyone’s sanity, *SpaceTow*), is, in short, a somewhat simplified clone of *Thrust* [1], in which the player controls a spaceship which must fly into a cave, find a payload, and tow it out of the cave, whilst trying to not crash into the walls of the cave or allow the payload to hit the walls of the cave, whilst trying to not fall foul of the laws of physics. It probably counts as being of the ‘arcade’/’action’ genres.

Gameplay is relatively simple. The player can apply torque to rotate their ship clockwise by holding left or D, apply anticlockwise torque by holding right or A, and apply linear force in the direction their ship is currently pointing in by holding up or W. When the player’s ship is close to the payload, they can attempt to start towing it by pressing space; if the ship is close enough, a mostly rigid connector will be created between the ship and the payload, and the payload will unfreeze, allowing it to be towed by (and potentially interfere with the physics of) the player’s ship.

|  |  |
| --- | --- |
| Diagram  Description automatically generated | Graphical user interface  Description automatically generated |

Figure : A couple of annotated screenshots of 'A Scientific Interpretation of Daily Life in the Space Towing Industry Circa 3052 CE'

Whilst *Thrust* had 6 levels (which looped with variations on them), *SpaceTow* currently only has 3 levels (ending after the third one), so there isn’t quite as much replayability. However, there is a relatively simple process for creating new levels. One merely needs to define the static level geometry (taking advantage of a factory method to convert a flat collection of CrappyBody objects into a QuadTree data structure), a background image, and the start positions of the ship/payload, edit the ‘levels’ enumeration to add a new member to it capable of generating that level (following the same process as how the other members of that enum generate the other levels), and that’s it. However, level 3 does differ a bit from the other two levels (and significantly from *Thrust*) by incorporating some unexpected ergodic literature. Other omissions from *Thrust* include the enemies, combat, barriers, and a hard limit on fuel (whilst *SpaceTow* does include a fuel counter, I wasn’t sure how much fuel would be a reasonable amount to expect a player to need in a level, so I decided to leave it as a counter of ‘total fuel used’ so a player can try to minimize it if they want to). Additionally, whilst a level of *Thrust* is completed by flying up, away from the level, the levels in *SpaceTow* are fully enclosed, but are instead won by bringing the payload past an invisible ‘finish line’, near where the player started (where, instead of flying away, the player is told that they won, as they watch the connector between the ship and the payload break, and the laws of physics taking control of the movements of both).

TODO: screenshot of *Thrust* for comparison

# Technical Issues

## The physics engine – CRAPPY

To be completely blunt, the main reason why I chose to implement my own physics engine from scratch instead of taking the sensible option of using JBox2D was simply because I wanted to use the name ‘CRAPPY’ (Cool Realism-Adjacent Physics Package, Y’know?) because I thought that giving it that name would be funny.

That said, it does do a few things that JBox2D doesn’t really do, such as supporting double-precision floating point numbers (instead of mere floats), offering immutable vectors (avoiding any unexpected side effects of some vector arithmetic) whist still providing mutable vectors (for people who may be concerned about garbage collection), and plenty of interfaces (for any programmers who may prefer using interfaces to using the classes themselves).

Despite this, it does still have several limitations, such as not supporting compound shapes, not having any joints besides an elastic connector, no raycasting, and, overall, a lot less polish.

## Features of CRAPPY used by my game

The f

## Physics principles used

## Bugs and unfinished features

## Parameters chosen

# Bibliography

|  |  |
| --- | --- |
| [1] | J. Smith, *Thrust,* Superior Software, 1986. |