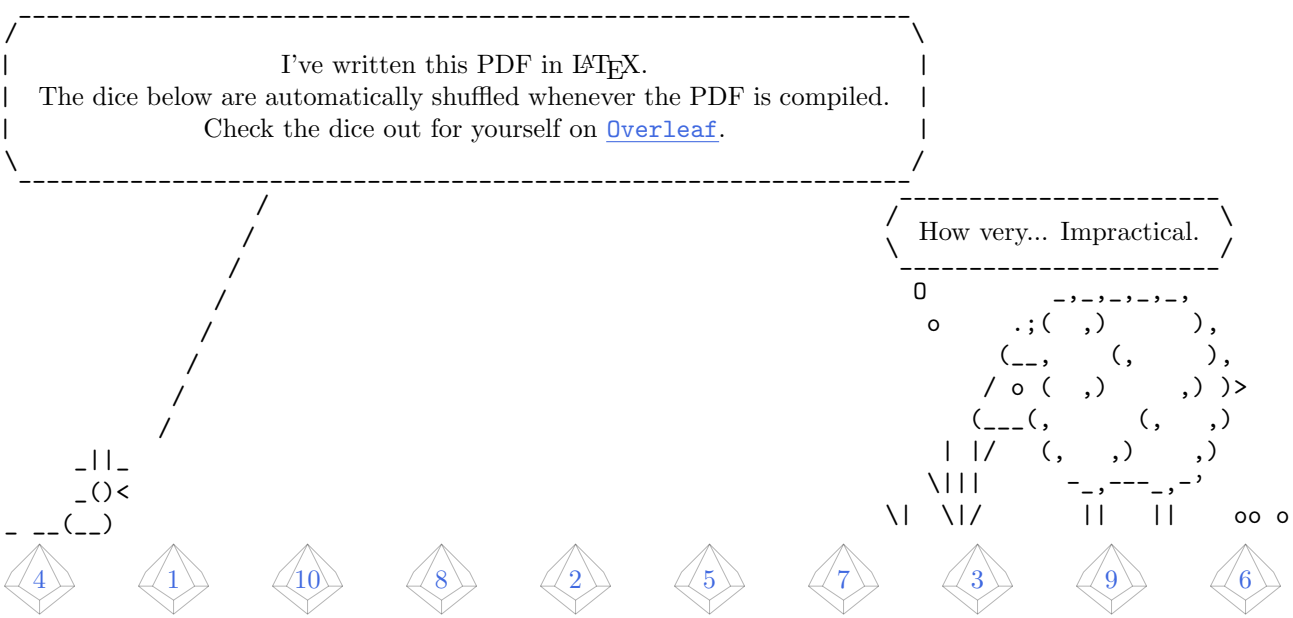
Joseph A. Binns

WORK SAMPLE

for

GAME PROGRAMMING

March 2022



Hi–!

BAA'–!

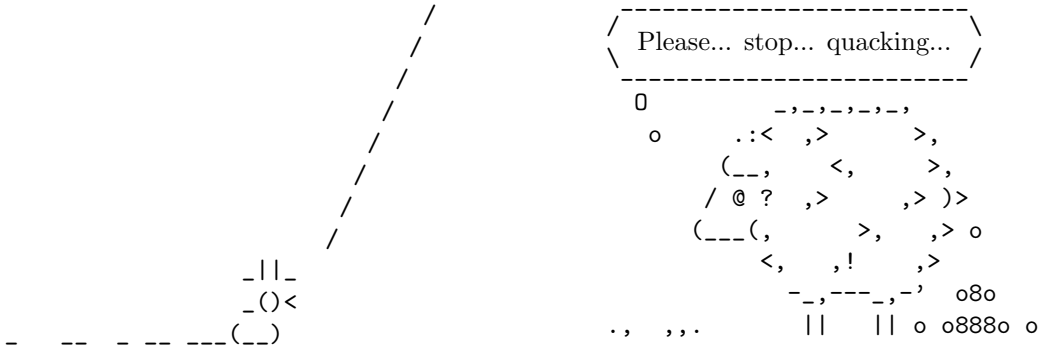
I'm Joe, a British game programmer and designer, working and studying in Sweden. I'm soon to graduate from Lund University with a BSc in Theoretical Physics. Alongside studies, I have leveraged my knowledge of maths and physics to create robust and reliable tools, games and STEM inspired simulations through Unity and custom game engines.

Woah buddy! Slow down th'–

Almost five years ago, I was presented with a video explaining how an artificial neural network had been trained with a genetic algorithm to play Super Mario World. I was hooked! That initial curiosity has continued to grow ever since; I am now writing my bachelor's thesis on the topic of neural networks.

...

Always eager to learn and find better ways of doing things, I'm dedicated to utilising my skills to facilitate unforgettable gameplay experiences. Here you can see a selection of professional and educational projects that I've had the opportunity to work on, as well as some of the personal fun-things that have led me here. Everything has been achieved whilst studying full-time; I am now looking for the opportunity to focus fully on game programming.



Personal

Stylised Character Controller

August 2021 — present

Collaborators: Clara Summerton (Sound Effects).

GitHub itch.io YouTube

Language C# Language ShaderLab

A stylised physics-based character controller made in Unity 3D. Before you read on, get a hands-on feel for the project over at itch.io, or watch the demo. Additionally, please take a few minutes to listen to my discussion about oscillators in game development!

The character controller is based on a floating capsule approach devised by Toyful Games for Very Very Valet. From the team's development blog, the various techniques for the movement are outlined and explained with snippets of code. However, the provided code was largely incomplete – partially due to it being tied up in the complex other workings of Very Very Valet. This lead me, with Toyful Games' permission, to develop and share my own modular take on the floating capsule approach. With the addition of some additional stylisation, the Stylised Character Controller project was conceived.

Features

- Physics-based character controller.
- Squash and stretch, which is a principle of animation, makes the character's movements appear more playful and less rigid.
- Dithered silhouettes appear on the character when obscured from view, letting the player know where they are at all times.
- Top down blob shadows on characters make 3D platforming feel sharper, and they look great when combined with Unity's inbuilt environmental shadows.
- Dust particles appear when characters move, making the character feel more alive.

Motivation

The motivation to start this project came after implementing the physics-based character controller into a game prototype of my own. I had been struggling for a long time to complete a project, since University was taking priority. With heaps of scrapped projects accruing, I felt a strong desire to just get something out there. So I took what I already had, polished it up, and released it in the hope that other game developers may find it useful. Enjoy!

Raymarch Explorer

April 2019 — September 2019

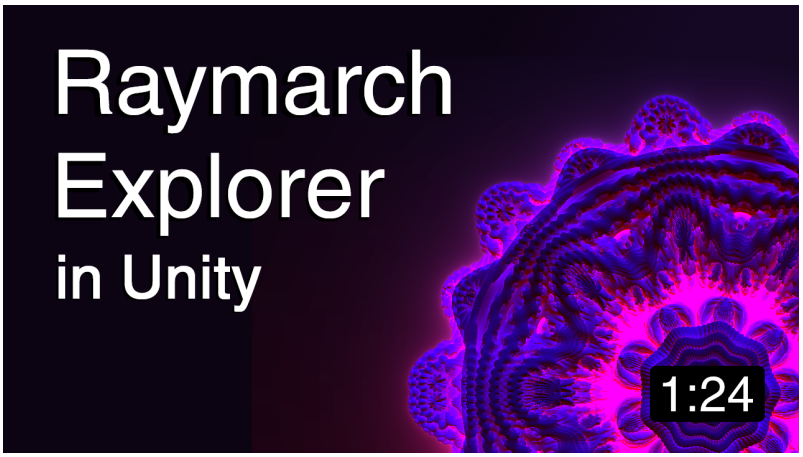
For this project, I familiarised myself with graphical programming and ray-marching techniques.

Unity build

GitHub YouTube

Language HLSL Language C#

Before you read on, check out the results on YouTube!



Raymarch Explorer is a real-time 3D fractal-explorer developed in Unity. The program prototypes a method for deep-zoom functionalities in which minimum distance values are passed from compute shader to C# script. The minimum distance value is then used to determine the fractal's level of detail and the camera's velocity; giving the effect of zooming into the fractal surface.

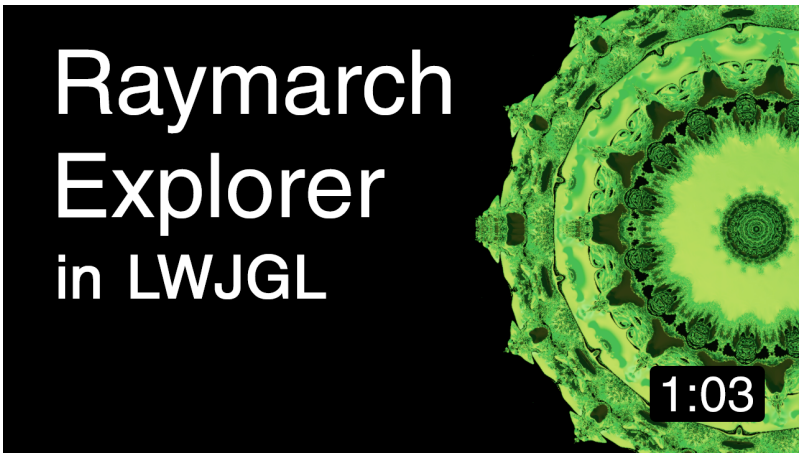
The depth of the fractal is restricted by data-type precision. In order to achieve true deep-zoom functionality, arbitrary precision arithmetics on the Graphics Processing Unit is required. Unfortunately, this task does not seem well suited to Unity due to inbuilt limitations on the data-type of the transform component. The project, which was originally based on Sebastian Lague's ray-marching-in-Unity repository, has therefore migrated to the Lightweight Java Game Library (LWJGL).

Lightweight Java Game Library build

GitHub YouTube

Language GLSL Language Java

Before you read on, check out the results on YouTube!



Raymarch Explorer is a real-time 3D fractal-explorer developed in a custom-built engine. The custom-built engine is developed from LWJGL for the Open Graphics Library (OpenGL).

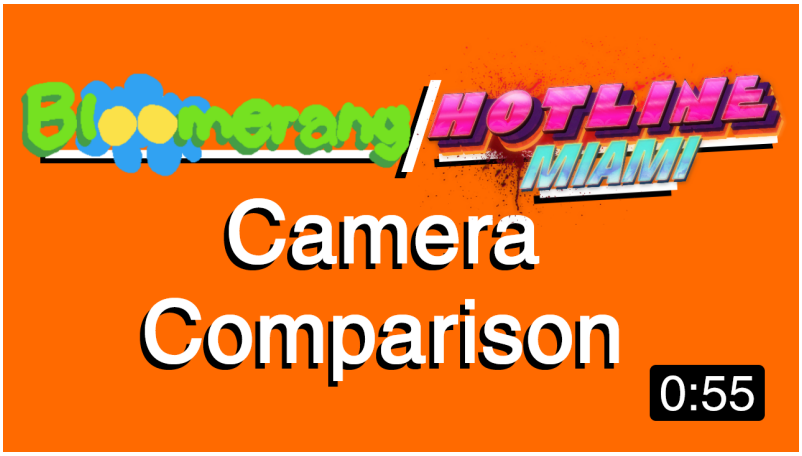
Bloomerang Game Maker's Toolkit Game Jam August 2019
Collaborators: Seelocanth (Art, Sound Effects & Music).
GitHub itch.io
Language C#

The garden is swarming with angry bees! Luckily, you've got your trusty 'Bloomerang' to help ward the bees off. Be careful though; You only get one shot to take enemies out. If you miss, the Bloomerang won't come back, and you may be finished!

Bloomerang is a procedural, top-down roguelike. This game was conceptualised and made in 48 hours for the Game Maker's Toolkit Jam (GMTK) in 2019. The theme of the jam was 'only one'.

Post-mortem

I was inspired at the time to attempt a 'Hotline Miami' style of aiming. This is where, to an extent, the camera follows the cursor's position. However, I was a little disappointed with the results. Others gave feedback that it didn't quite feel right. I agree that it does feel a little un-intuitive – even disorienting. But why? A short side-by-side video was put together in order to find out what went wrong. Check out the comparison on YouTube!



Did you spot it? Well, I didn't at first. The main difference I wish to highlight is only seen in the behaviour of the mouse when close to the player. It's not very well illustrated in the video, so I recommend getting a feeling for it yourself over at itch.io.

I implemented a minimum-distance radius that the cursor can be to the player. The idea being that it would prevent players from throwing the boomerang little to no distance. Though this seemed a good idea to me – as why would a player ever want to throw the boomerang to no effect? It had an unintended consequence; The player's control over the cursor had become obstructed. When the player moves the cursor over the character, they expect it to continue in a straight line. However, the updated cursor moved in a minimum-distance radius around the character. This resulted in un-predictability, most noticeable in the intense situations where it mattered the most – when enemies are too close for comfort. The message is clear; design decisions, particularly those risking obstructing the player's freedom, must be iterated on – regardless of good intent.

Dare to Dance Ludum Dare April 2019

GitHub itch.io Ludum Dare

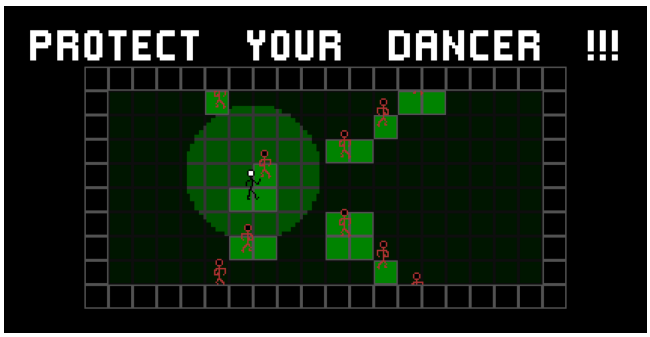
Language C#

The floor's come alive with zombies; Dance on the beat to ignite their disco spirit!

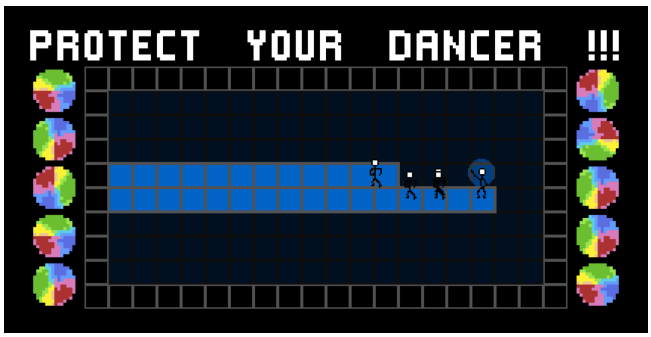
Dare to Dance is a fast-paced, disco-themed rhythm game. This game was conceptualised and made in 48 hours for the Ludum Dare 44 game jam. The theme of the jam was 'your life is currency'. Before you read on, get a hands-on feel for the project over at itch.io!

Post-mortem

For a first attempt at a game jam, I am pleased with Dare to Dance, not least of all for it's core concept and visual simplicity. The core concept was that the player can only attack on the beat, which was synced to the beats-per-minute of the music. I think I did well, with my toolkit at the time, to emphasise the beat itself using visual queues; Such as a closing spotlight and flashing disco-balls.



1. Don't get touched...



2. Strike on the beat!

However, there is certainly a lot of room for improvement primarily based around the controls being hard to use and unforgiving. For instance:

- The timing window for attacking should be increased, for greater leniency.
- The player should not be required to move while attacking.
- The last direction of movement should automatically be used as the attack direction.

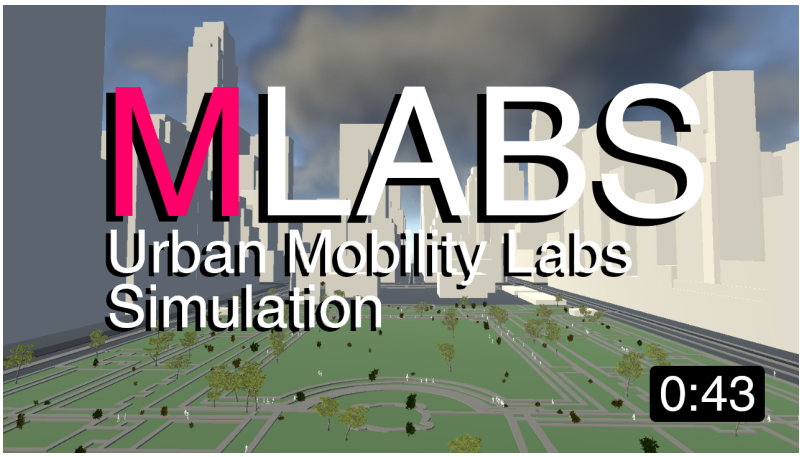
Beyond remedying the controls, a lack of feedback on missed attacks is resulting in a frustrating, un-supervised learning experience for the player. The game also has a lot to gain from additional layers of complexity. The life of the player should not be restricted to just the original dancer, but should instead be spread across all the dancers you have acquired. This would mean that the game would continue as long as a single dancer remains. I would also like the player to be provided with more information in regards to enemy spawn locations, e.g. with enemies rising out of certain highlighted floor tiles. Alas, time was of course the main limiting factor. However, I am sure the end result would have more to show in the same time, had I embraced rather than shy-ed away from putting prototypes of the game in front of others, for their feedback. Lesson learnt!

Professional

MLabs Simulation MLabs, London — July 2019 — August 2019
Collaborators: Dilan Patel (Pedestrians).
YouTube Letter of Recommendation
Language C#

Fresh out of school, with a few projects to show, I was fortunate enough to acquire a software engineering position at a London-based start-up. Hired for my experience in Unity, I worked within a small team. We spent the summer developing a realistic, real-time, urban environment simulation. The simulation was designed for use by an automated transport planning system.

I implemented access to OpenStreetMap's data via the Mapbox Software Development Kit for Unity. The data provided by OpenStreetMap's for the simulation granted instant access to the open-source data required to build urban environments from across the globe. I then used meta-data from OpenStreetMap's to facilitate decorating the world with parks and pavements. Finally, I created a day-and-night weather cycle. Having established the core framework for future developments, I left the project to begin my BSc in Theoretical Physics at Lund University. Check out the initial build of the project on YouTube!

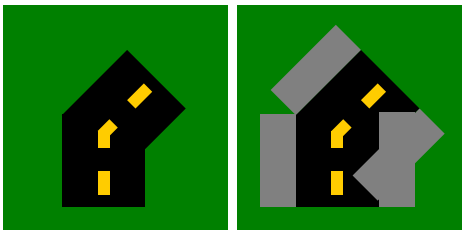


Features

- Buildings, roads, and pavements generated in real-time from world-wide OpenStreetMap's data.
- Pedestrian pavement path-finding and collision avoidance.
- Parks and grasslands, scattered with trees and bushes.
- Performance efficient clouds in a day-and-night weather cycle.

Post-mortem

The buildings and roads provided by OpenStreetMap's alone are insufficient to truly re-create an urban environment – without some impressionism. For example, how could we create procedural pavements for people to walk on? Since the metadata for roads was limited to just the carriageway type, some general assumptions were needed, such as motorways not having pavements. The problem then became how to create the pavements themselves. Initial naive attempts to take each segment of road, make two raised copies, and shift them to either side of the road did work – except on the slightest of curves.



Problematic pavements

Procedural mesh pavements

I had hope of a better solution involving procedural mesh creation – something which I didn't have any experience in at the time. After doing some research and prototyping, I was ready to give it all a try. I made a new game object for the pavements, and generated it's mesh from a fixed cross-section consisting of pavements separated by the width of the road. I'm glad I tried, because this alternative approach worked great!

Educational

BSc Theoretical Physics Thesis Lund University January 2022 — June 2022
Supervisors: Patrik Edén, Mattias Ohlsson.
Collaborators: Rasmus Sjöö, Alexander Degener, Oscar Bolinder.
Letter of Recommendation
Language Python

Artificial Neural Networks: Testing the Weight Decay Regularisation Strength
Artificial Neural Networks are prone to over-fitting. Over-fitting is when a network trains too exactly to a data set, resulting in the particular noise of the data being learnt. Learning noise causes the network to generalise poorly to other data sets, regardless of being from the same statistical distribution. What we aim for the network to learn instead is this underlying statistical distribution of the data.

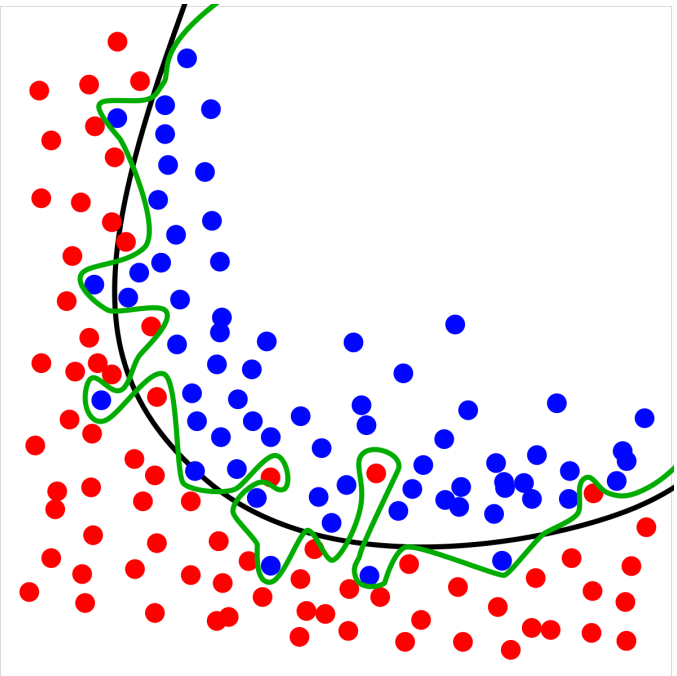


Illustration of Overfitting
By Ignacio Icke, licensed under Creative Commons.

Weight-decay is a common technique used to reduce over-fitting. Weight-decay acts to minimise complexity by penalising excessively large neural connections. However, there is a fine line to be drawn. The amount of weight-decay is controlled by the weight-decay-strength, too much of which, and large neural connections will be penalised so much that all connections in the neural network will tend to zero during training; Rendering the neural network useless. The current standard for finding a good weight-decay-strength is for it to be varied through trial-and-error.

My thesis will test theoretical arguments for the weight-decay-strength. If a hypothesised theoretical argument is successful, it would mean that the weight-decay-strength can be set without trial-and-error. As the questions cannot be studied sufficiently in standard Keras code, the first part of the project has been in coding our own Artificial Neural Network. I have been responsible for the stochastic data generation, statistics calculations and leading the team's large-scale neural network training via remote access to computer clusters.

This will be the longest running programming project, on a group of this size, in which I have been involved. I have greatly enjoyed the opportunity to collaborate on this project; From which we have manifested an environment that is bursting with ideas. I look forward to similar large team-projects in the future.

Machine learning is becoming increasingly valuable to all kinds of disciplines, not least of all games. In my career, I have a particular interest in working on the continuously widening overlap between the fields of games and machine learning. I get particularly giddy when thinking about some of the incredibly innovative and novel areas in which game programming will benefit from machine learning in the near-future. Whether it be in making NPCs 'come to life', streamlining game-testing or something else entirely; You can trust that I'll be getting involved!

Disease Projection using Ordinary Differential Equations

Lab Report

Language Python

The main purpose of this project was to explore and experiment with a variety of models involved on the topic of disease spreading. As the starting point of the mathematical models focused on how an infected and susceptible population changes over time, the equations would need to be solved in order to yield the number of infected at a given time. As this is a computational physics course, I was directed to computationally solve the equations! Computational solutions are often practical, but introduce new sources of errors. As such, the project also served as a lesson on statistical error calculations, as well as error reduction through improved techniques and solvers. Further additions were made to the model, such as the introduction of vaccinations and the movement of populations between discrete locations. Once the appropriate toolkit had been assimilated, an appropriate model for a chosen case-study disease outbreak was simulated – HIV in western Africa in the middle of the 20^{th} century.

This project improved my competence as a games programmer, though as noted in the lab report, my avoidance in Object Oriented Programming in Python started to become a hindrance as the complexity of the project increased. This was something which I made sure to not let happen again in the latter Computational Physics projects! Such deep studies into algorithms such as Runge-Kutta have made me increasingly confident in seeing the applicability of such algorithms, and made me comfortable in their implementation.

Image Restoration using Partial Differential Equations

Collaborators: Joshua Greaves, Daniel Larsson Persson.

GitHub YouTube Presentation Slides

Language Python

Equations originating from physics have recently found their way to other areas. One, possibly surprising, application is that of restoring art or images. To understand how Partial Differential Equations (PDEs) from physics can help with image restoration, consider a greyscale image of your own choice. Due to some unwarranted graffiti, let us assume that a piece is missing (black region). Can we fill in this missing region without any information as to what is missing? This may seem like a hopeless task, but PDEs are here to help!

PDE-based methods for image restoration are based on propagating the information (typically, colour-specific intensity values and gradients) at the boundaries of the missing region inwards. The propagation is performed by solving a partial differential equation for the missing regions' given boundary conditions. Check out the results on YouTube!

I really enjoyed collaborating on this project, where I created the initial code logic, and later took responsibility for handling user inputs and finding innovative ways of improving the algorithms results through a new technique. This new technique consisted of breaking the regions to be solved into individual sections; The border colours of which were then averaged over to create an appropriate initial fill colour for each section. This was found to both speed up solving and greatly improve results for large pieces of graffiti. Though we just touched the tip of the iceberg in terms of image restoration techniques, the project demonstrated how useful physics-based equations can be in novel applications, such as graphical programming.

Vicious Walkers
Lab Report YouTube
Language Python

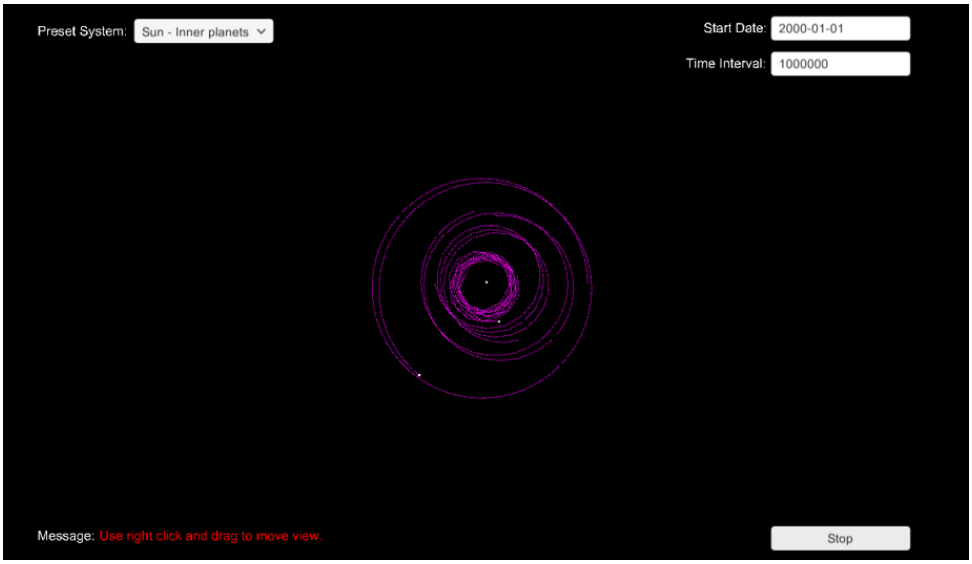
The main purpose of this project was to explore and analyse various scenarios surrounding random walkers. The number of dimensions, initial conditions and a variety of other factors were used to facilitate these experiments. The results were then compared to theoretical expectations.

I found this deep dive into stochastic systems to serve as a powerful parallel to a focused analysis of dynamic systems, or NPC behaviour in games. The study also solidified my comfort in working with Object Oriented Programming in Python, which came in a lot of handy as the vicious walkers became increasingly complex! Check out the results on YouTube, and read my analysis in the lab report.

N-body simulator Forest Hill School, London August 2017 — March 2018
Design Document & Development Log
Language C#

A real-time, N-body gravitational simulator and graphical user interface in Unity. The program accesses planetary data from the NASA Horizons database via Telnet. Various complexity-reducing algorithms have been utilised; Optimising the large-scale physics-based simulation.

Excited by the utilisation of programming in support of physics, this project initially started as an old blog article and excel spreadsheet of mine. The project is a follow up inspired by a great Feynman lecture on Newton's Laws of Dynamics. Read about how I designed and tested the program, as well as the hurdles I had to overcome in it's development, in the design document & development log.



Screenshot from N-body simulator