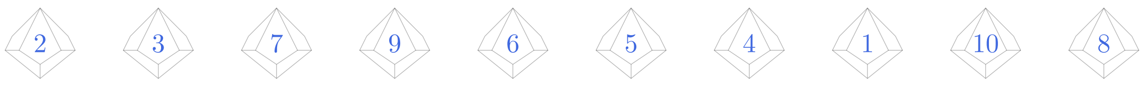
J O S E P H A L E X A N D E R B I N N S

Work Sample for Game Programming

GAMES

2022



I've written this PDF in LaTeX.
These dice are automatically shuffled whenever the PDF is compiled.
Check the dice out for yourself on `Overleaf`.
...
How impractical!

Preface

Blah blah blah I am awesome and the right candidate, please accept me.

add preface

Personal

Stylised Character Controller

August 2021 - present

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. Alternatively, watch the demo on YouTube!

The character controller is based on the floating capsule approach devised by Toyful Games for Very Very Valet. In a video from the team's development blog, the various techniques for the movement are outlined and explained. The video also provides snippets of code, though incomplete in places. The source code was not provided by Toyful Games due to it being tied up in the complex other workings of Very Very Valet. This project contains a fan-made pure re-creation of their physics-based character controller.

Additional stylisation inspired by discussions found in Toyful Games blog posts on character animations and shaders and effects are also included. These implementations exist from a personal desire to have them in my own projects. The project makes use of Unity's Universal Render Pipeline (URP) to facilitate some of these graphical features.

Features

- Physics-based character controller.
- Squash and stretch on the character makes motion appear more fluid and bouncy, it is a principle of animation.
- 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 shadows on the environment.
- 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. University was taking priority. And with heaps of scrapped projects accruing, I felt a strong desire to just get something out there. So I took what I already had, and released it in the hope that other game developers may find it useful. Here it is!

Raymarch Explorer

April 2019 - September 2019

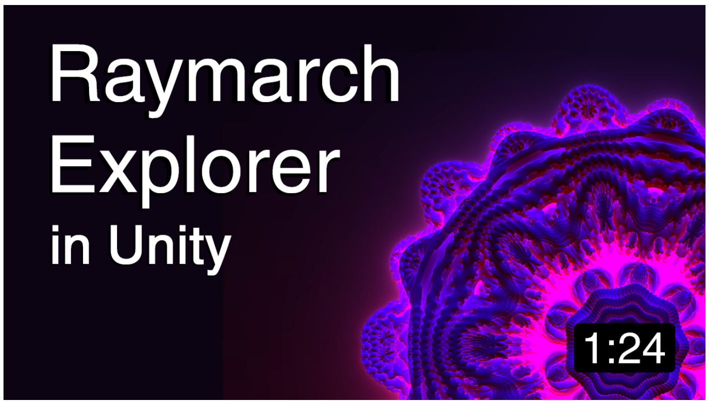
For this project I familiarised myself with graphics programming and ray marching techniques.

Unity build

GitHub YouTube

Language HLSL Language C#

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



Real-time 3D fractal explorer in Unity. Prototyping a method for deep zoom functionalities by passing minimum distance values from the compute shader to a C# script. The minimum distance value is used to determine appropriate velocities and level of detail, such as to give 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 (GPU) 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 has therefore been moved to the Lightweight Java Game Library (LWJGL).

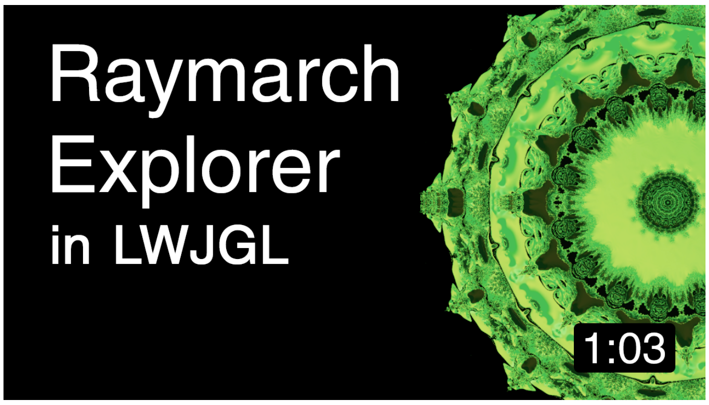
Developed from Sebastian Lague's Ray-Marching repository.

Lightweight Java Game Library build

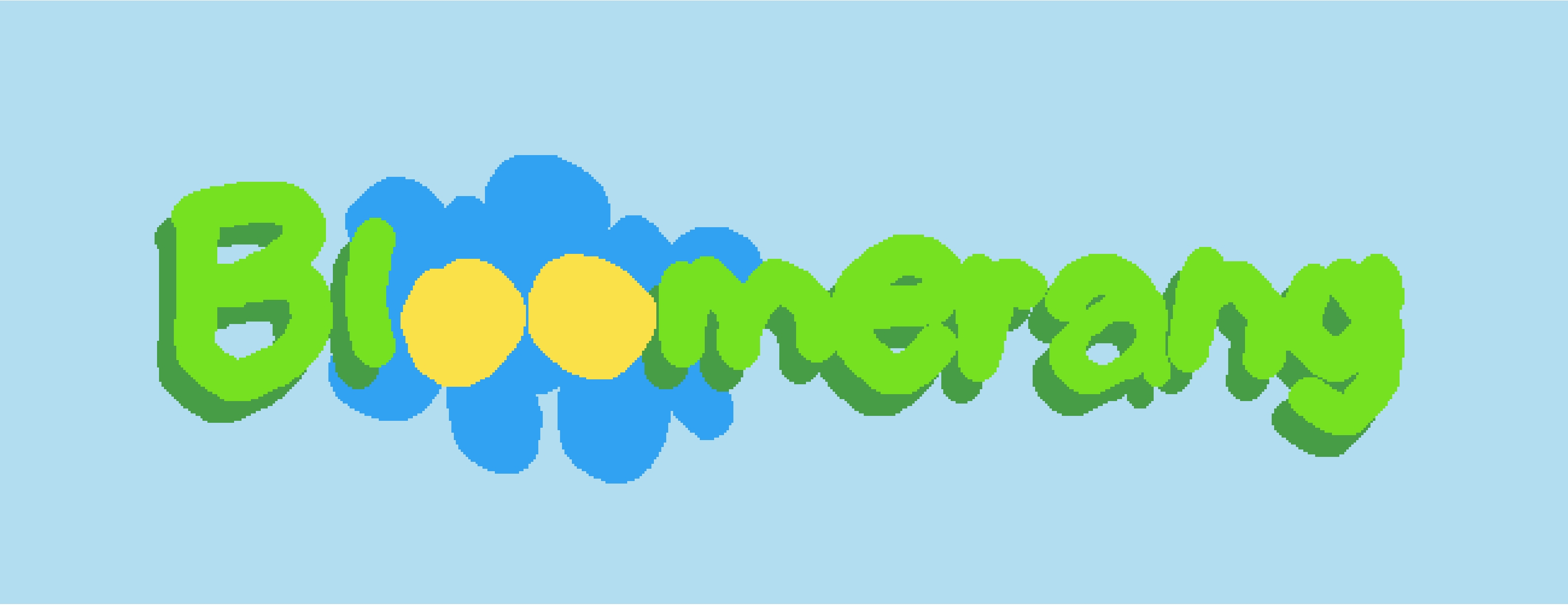
GitHub YouTube

Language GLSL Language Java

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



Real-time 3D fractal explorer in a custom-built engine. The custom-built engine is developed from the Light Weight Java Game Library (LWJGL) for the Open Graphics Library (OpenGL). The depth of computed fractals is restricted by data-type precision. In order to achieve true deep zoom functionality, Arbitrary Precision Arithmetics is required. This project explores arbitrary precision arithmetics on the Graphics Processing Unit (GPU).



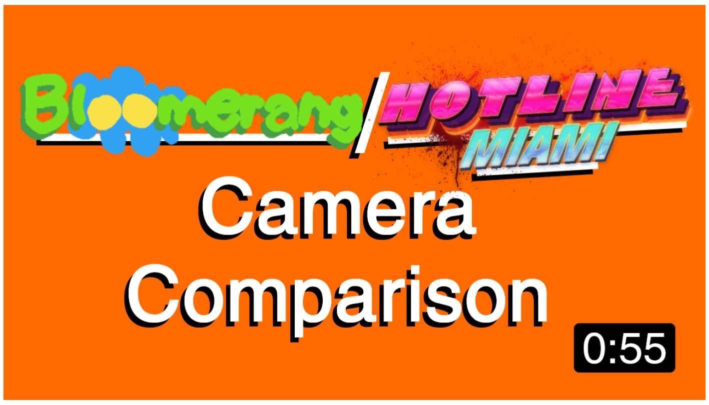
Bloomerang Game Maker's Toolkit Game Jam August 2019
Collaborators: Seelocanth (Art & Music).
GitHub itch.io
Language C#

The garden is swarming with angry bees! Luckily, you've got your trusty 'Bloomerang' to help ward them 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) 2019. The theme of the jam was 'only one'. Before you read on, get a hands on feel for the project over at itch.io!

Post-mortem

I was inspired at the time to attempt a 'Hotline Miami' style of aiming. This is where the camera follows the cursor position, to an extent. I was however a little disappointed with the results. Others gave feedback that it didn't quite feel right. I agree that it does feel to be a little un-intuitive and disorienting. I wanted to determine why this was. A short side-by-side video was put together in order to analyse why it worked poorly in Bloomerang compared to the original inspiration. Check out the comparison on YouTube!



Did you spot it? 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 radius that the cursor can be to the player. The idea was that it would prevent players from throwing the boomerang little to no distance. Though this seemed to be a good idea to me - as why would a player ever want to throw the boomerang to no effect? It means the players control over the cursor is obstructed. While the player expects the cursor to continue in a straight line as it moves over the character, it now moves in a minimum-distance radius around the character. Resulting in an un-predictable, fast moving cursor. The message is clear; design decisions that are at risk of obstructing the control of the player must be tested and iterated, regardless of good intent.

Dare to Dance Ludum Dare

April 2019

GitHub itch.io Ludum Dare

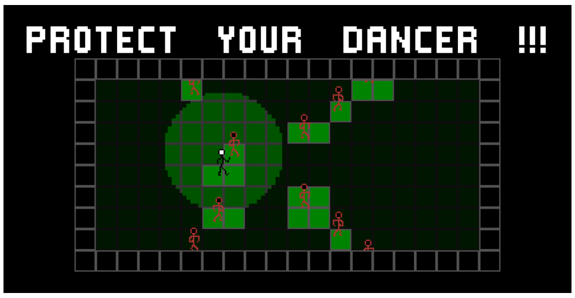
Language C#

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

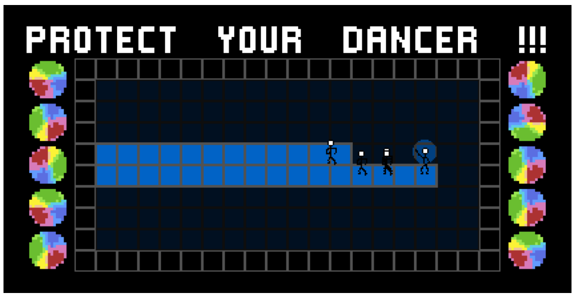
Dare to Dance is a Fast Paced Disco themed Rhythm Game. This game was conceptualised and made in 48 hours for the Ludum Dare 44 game jam. The theme of the jam was 'your life is currency'. Before you read on, get a hands on feel for the project over at itch.io!

Post-mortem

For a first attempt at a game jam, I am pleased with Dare to Dance. I especially enjoy it for it's visual simplicity. The beat was synced to the Beats Per Minute (BPM) of the music, and the beat itself was emphasised by both the closing spotlight and flashing disco balls. However, there is a whole lot of room for improvement.



1. Don't get touched!



2. Strike on the beat!

The controls may be hard to use and unforgiving. The timing window for attacking should be increased, for greater leniency. The player should not be required to move while attacking. Instead, the last direction of movement should automatically be used as the attack direction. At the moment, a lack of feedback when attacks are missed is resulting in a frustrating, un-supervised learning experience for the player. Beyond polishing the player controls, the game has a lot to gain from additional layers of complexity. The life of the player should not be restricted to the single central dancer, but spread across all the dancers you have acquired. Such that you survive as long as at least a single dancer remains. I would also like enemy spawn locations be illustrated. For example, with enemies rising out of certain highlighted floor tiles. The fundamental premise also needs to be built upon. For example, certain dance moves could be required to be performed in order to progress. These could be performed by fluidly moving over a pattern of highlighted tiles.

Professional

MLabs Simulation MLabs, London July 2019 – August 2019

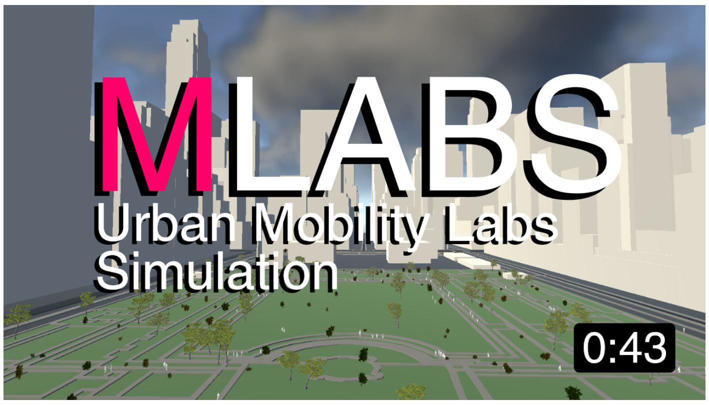
YouTube

Language C#

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

The simulation accesses OpenStreetMap data via the Mapbox Source Development Kit (SDK) for Unity. Providing the simulation instant access to open source urban environments from across the globe. Having established the core framework for future developments, I was set to commence my BSc Theoretical Physics.

Check out the initial build on YouTube!



Features

- Buildings, roads, and pavements generated in real-time from OpenStreetMaps data from anywhere in the world.
- Pedestrian pavement path-finding and collision avoidance.
- Parks and grasslands, scattered with trees and bushes for added depth.
- Performance efficient clouds.
- A day-night cycle.

Educational

BSc Theoretical Physics Thesis Lund University January 2022 - June 2022

Supervisors: Patrik Edén, Mattias Ohlsson.
Collaborators: Rasmus Sjöö, Alexander Degener, Oscar Bolinder.

Language Python

Artificial Neural Networks: Testing the Weak Decay Regularisation Strength

Artificial Neural Networks (ANNs) are prone to overfitting. Overfitting is when a network trains too exactly to a data set. Such that the particular noise of the data is unknowingly being learnt. What we aim to learn instead is the underlying statistical distribution of the data. Learning noise causes the network to generalise poorly to other data sets, regardless of being from the same statistical distribution. Weak decay (L2) is a common regularisation technique used to reduce overfitting. L2 acts to minimise complexity by penalising excessively large weights. However, there is a fine line to be drawn. Too much L2, and large weights will be penalised so much that the model will tend to set all weights to zero. The L2 regularisation strength is used to find this fine line.

include a picture, to facilitate the description

The proposed topic is to test theoretical arguments that determine the weak decay (L2) regularisation strength. If it works, it would mean that the parameter can be set without trial-and-error. It could also mean that it is best to have different strengths for different nodes, and even bias weights. The questions cannot be studied in standard Keras code. And so the first part of the project is to code my own ANN.

Connect to games. I.e. in explanation of overtraining. Feel free to add an illustration

Computational Physics Lund University August 2021 – October 2021

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 the infected and susceptible population **change** 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. And as such the project also served as a lesson on error estimation, as well as error reduction through improved techniques and solvers. Further improvements were made to the models, 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 20th century.

Read the lab report.

Image Restoration using Partial Differential Equations

Collaborators: Joshua Greaves, Daniel Larsson Persson.

GitHub YouTube

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 in image restoration, consider some grayscale image of your own choice. Due to graffiti painters, let us assume that a piece is missing (black region). Can we fill in the missing region without any information of 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!

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 compared to theoretical expectations.

Check out the results on YouTube!

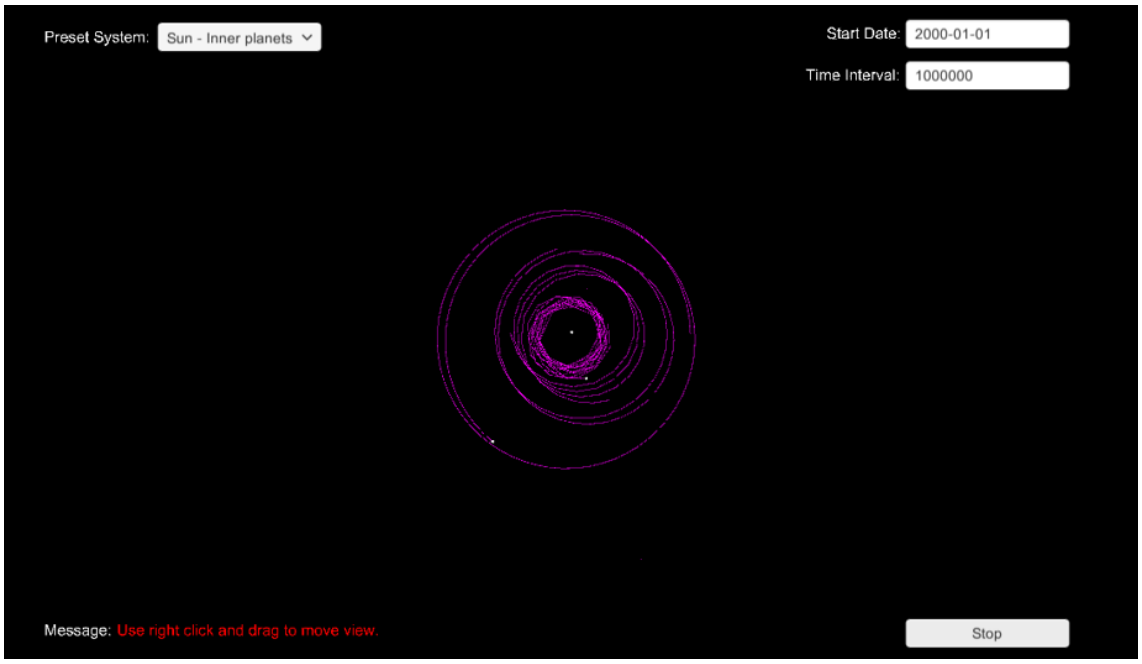
Read the lab report.

***N*-body simulator** Forest Hill School, London August 2017 - March 2018
Development Log

Language C#

A real-time, N-body gravitational simulator and Graphical User Interface (GUI) in Unity. The program accesses planetary data from the NASA Horizons database via Telnet. Various complexity reducing algorithms have been utilised, to optimise 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. The project is a follow up inspired by a great Feynman lecture on Newton's Laws of Dynamics.



Preset System: Sun - Inner planets

Start Date: 2000-01-01

Time Interval: 1000000

Message: Use right click and drag to move view.

Screenshot from N-body simulator

Read the development log.