******

**CS4343 Game Development Project**

**Hunny Hunt**

***Final Report***

|  |  |
| --- | --- |
| Benedict Lim | U077103A |
| Liew Jia Hao | U086949W |
| Mark Chen | U087067B |
| Neo Qi Yang | U080305L |

Table of Contents

[Executive Summary 3](#_Toc290410068)

[Breakdown of Team Contribution 4](#_Toc290410069)

[Game Design 5](#_Toc290410070)

[Rationale 5](#_Toc290410071)

[Technical Challenge Summary 6](#_Toc290410072)

[Project schedule 7](#_Toc290410073)

[Liew Jia Hao 8](#_Toc290410074)

[Main Contributions 8](#_Toc290410075)

[Verlet Integration 8](#_Toc290410076)

[Interfacing with Unity3D game environment 9](#_Toc290410077)

[Time constraint 9](#_Toc290410078)

[Side contribution 9](#_Toc290410079)

[Mark Chen 10](#_Toc290410080)

[Main Contributions 10](#_Toc290410081)

# Executive Summary

Hunny Hunt is a physics exploratory game that seeks to examine the possibility of implementing soft body dynamics with an additional fluid simulation for soft bodies and particles in Unity3D.

On top of the physics simulation, our team would also have to implement an appropriate representation of our Hunny blobs through the use of Shaders and procedural mesh deformation.

Finally, the ultimate goal of our project is also to contribute back to the Unity3D developer’s community by releasing our physics source code based on Verlet Integration so that the community would be able to implement their own soft bodies and fluid simulations easily in their own games as well.

// Insert pictures.

# Breakdown of Team Contribution

|  |  |
| --- | --- |
| Member | Contribution |
| Liew Jia Hao | * **Anything pertaining to physics calculation and simulation.** * **Interfacing between in-house physics codes with Unity game world.** |
| Mark Chen | * **Physical representation of Hunny Blobs** * **Shaders programming** * **Procedural mesh deformation** * **Middle man that understands both physics code and level code for optimal collaboration** |
| Benedict Lim |  |
| Neo Qi Yang |  |

# Game Design

Hunny Hunt is a puzzle game developed in Unity3D game engine that is somewhat similar to puzzle game genres such as Mercury Meltdown and Loco Roco. Players would have to manipulate various game objects within the game in order to guide the Hunny Blobs into its final goal destination – “The Hunny Pot”.

The challenge lies in figuring out the optimal stage configuration to transport as much Hunny blobs as possible to its final destination. The game ends when ? or if the required amount of Hunny has been collected.

The game is designed to be as easy to play as possible, without having the player to go through any tutorials at all. Tooltips are shown upon hovering over each game object to allow for an easy learning curve.

The novelty of our game lies in the fact that there hasn’t been similar fluid based puzzle game. Players would enjoy and have fun playing with the realistic fluid like movement of the Hunny blobs while thinking of a good solution for each stage.

# Rationale

Our group has decided to implement a fluid based puzzle game as it hasn’t been previously done before. We thus seek to explore the feasibility of having such kind of game genre and its playability. Our findings would thus be valuable to developers looking to develop similar game genres in the future.

# Technical Challenge Summary

There are 4 main technical challenges experienced by our team in the development of our game, namely: Physics calculation, Visual representation, interfacing with Unity’s game world and level design.

|  |  |
| --- | --- |
| Physics calculation | * All particle and body movement calculated based on Verlet Integration * Collision detection and collision response * Inter body attractive forces * Soft body * Smooth fluid movement |
| Visual representation | * **Shaders** * **MetaBalls** * **Procedurally generated mesh representation** |
| Interfacing with Unity’s Game world | * **Handshaking between Unity’s gameworld and our internal physics bodies.** * **Abstraction of our internal calculation, allowing others to manipulate physics bodies through the Unity editor** |
| Level design | * **Assets sourcing and management** * **Scripting of assets to manipulate Hunny Blobs** * **Scripting of assets for proper game play** |

# Project schedule

.// insert timeline.

# Liew Jia Hao

## Main Contributions

* Coded all related physics simulation and body movement based on Verlet Integration
* Collision detection
* Interfacing with Unity game world

## Verlet Integration

Verlet integration is an approach to physically based modeling which is well suited for interactive use such as in games. It allows for simulation of both cloth; soft and rigid bodies; and even articulated or constrained bodies using both forward and inverse kinematics.[[1]](#footnote-1)

The algorithm for Verlet Integration was developed originally for IO interactive’s game, titled Hitman: Codename 47. The algorithm is thus responsible for the movement of all the Hunny Blobs and deformation of the Hunny Blobs as well.

In summary, the entire verlet integration simulation code base could be broken down into the following technical challenges:

* A Verlet integration scheme.[[2]](#footnote-2)
* Handling collisions and penetrations by projection.
* A simple constraint solver using relaxation.
* Modeling our Hunny blobs as particles with constraints
* Collision engine with the ability to calculate penetration depths, along with a collision response to handle the penetration between bodies
* Scaling it to handle large amount of Hunny Blobs

## Interfacing with Unity3D game environment

There is a significant amount of challenge in implementing the algorithm describe in the research paper in Unity3D and also allowing it to interact with other Unity3D game object seamlessly.

This would thus give team mates the flexibility to create any form of Unity3D game objects within the Unity3D editor itself and yet being able to interact with the Verlet Integration physics body.

This abstraction allows the level designer to design the level without having to care about how the physics simulation is being implemented.

## Time constraint

Since the entire group could not proceed without the core physics code being up, I had to quickly research and read up on various fluid simulation methods being used in the game industry and implement one that best fit our overall game design goal at the earliest possible time.

Alternative implementation such as Smoothed-particle hydrodynamics (SPH) were tried out as well, but was found to be unsuitable for our needs.

## Side contribution

I have also tried a basic 2D MetaBall representation the HunnyBlobs, but it was found too laggy to be used.

# Mark Chen

## Main Contributions

* Exploring various representation including : 3D metaballs, 2D metaballs, shaders, meshes
* Integration between physics and level design.
* State machine ? // mention or keep quiet lol.
* Brought a 2nd ps3 controller to play SF4
* Created a hole in Media lab wall by constant 24/7 banging of head for shaders..
* Founder of Anti golam association 2011.

1. http://www.gotoandplay.it/\_articles/2005/08/advCharPhysics.php [↑](#footnote-ref-1)
2. http://www.fisica.uniud.it/~ercolessi/md/md/node21.html [↑](#footnote-ref-2)