College of Engineering and Informatics

B.Sc. (CS&IT) CT413 – Final Year Project 2019/2020

Project Definition Document

Student Name	Student ID	Project Name	Project ID
Irish Senthilkumar	16342613	Soft Body Physics	SR6

Table Of Contents

List Of Figures

List Of Tables

Glossary

UE4	Unreal Engine 4
AI	Artificial Intelligence
3DMS	3D Modelling Software
BVH	Bounding Volume Hierarchy

Chapter 1. **Introduction and Milestones**

1.1. Introduction

This project is centred on the development of an application demonstrating soft body physics. The application chosen for this project is a demolition derby game, where the player can crash their vehicle into other vehicles and observe the vehicular deformation. This project definition document outlines the research that was carried out prior to developing the project, and the findings of this research.

Soft body physics is a recent field in computer graphics that focuses on realistically defining the motion of soft bodies. At a computer graphics level, these are scene objects whose points do not maintain a fixed distance from each other. At the fundamental level, all scene objects are comprised of polygons, which in turn are made of vertices which are connected by edges. These polygons can be combined in different orientations to produce a mesh, and this mesh can be manipulated to simulate soft body physics.

This project definition document is broken down into 4 sections. Section 1 is this introduction. Section 2 outlines the physics behind soft bodies and the factors which affect their deformation. Section 3 describes the structural components of a car and the characteristics of these components which affect deformation. Section 4 outlines the physics engines and 3D modelling software that were considered for use in this project. Section 5 describes in detail the approached planned to implement soft body physics using the chosen physics engine and 3D modelling software.

1.2. Milestones

Milestone	Description	Due Date
Application Nature Decision	The nature of the application (video game or demonstrator application) is decided.	21st November 2019
Vehicle Suspension	Achieve realistic wishbone suspension simulation for the vehicle.	30th December 2019
Vehicle Steering	Achieve realistic steering for the vehicle.	30th December 2019
Mesh Collider Substitute	Create a performance-friendly implementation of mesh collider to use on soft bodies.	30th January 2020
Deformation	Complete deformation physics for a number of materials.	30th January 2020
Vehicle Completion	Complete the 3D models for the full car.	20th February 2020
Application Requirements Complete	Combine deformation physics and the car's 3D models to complete the requirements for the project.	25th March 2020

Chapter 2. **Principles of Deformation**

2.1. Introduction

In real life, no object is completely rigid. Under strain, an object will deform to absorb the force applied on it. An object can deform in many ways, but there are two types of deformation: elastic and plastic [1]. This section outlines the science behind deformation and discusses key concepts that will need to be accounted for in computer graphics in order to present a realistic soft body simulation.

2.2. Elastic and Plastic Deformation

For tensile stress (forces that pull on an object), elastic deformation is caused by low stress on an object, where a temporary change in shape is observed due to the molecular bonds being stretched but not broken [2]. The shape of the object returns to normal after the stress is removed. However, if a large enough stress is applied to an object, the object will undergo plastic deformation. At a microscopic level, this deformation results in the molecules breaking their bonds and sliding past each other. Elastic deformation can be represented using Hooke's law as this is similar to a spring system, but only when the elastic deformation is linear [1]. Plastic deformation is non-reversible, so the object will never return to its original shape when the stress is removed. The elastic and plastic deformation stages can be represented on a stress/strain curve. As seen in Figure 2.1, strain and stress are proportional to each other up to the yield strength. Elastic deformation is observed up to the yield strength point, and afterwards the object undergoes plastic deformation until the fracture point is reached. Upon reaching the fracture point, the object breaks apart. The ratio of elastic and plastic deformation is unique for all materials. Rubber bands have a large elastic region and a small plastic region, while glass has a small elastic region and a small plastic region.

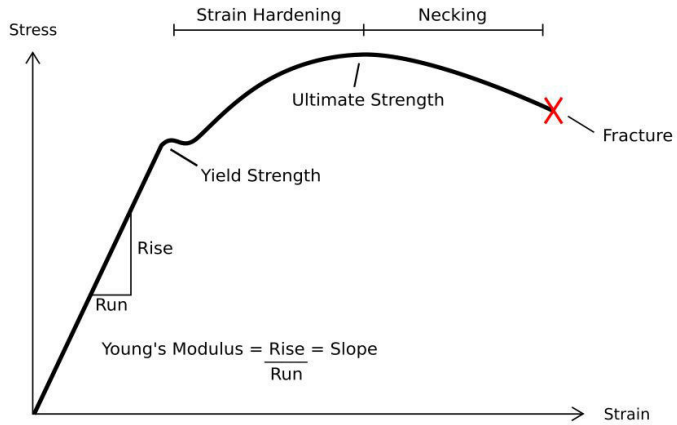

Figure 2.1: The stress/strain graph for an object.

Compressive stress (forces that push on an object) is the opposite of tensile stress, and objects display different behaviours under each type of stress. Some materials such as steel, can withstand high tensile stress and compressive stress. Other materials such as concrete, can withstand very high compressive stress but break apart during very little tensile stress [3]. When an object exerted to shear, bending, and torsion stresses, both tensile stress and compressive stress coexist.

Chapter 3. **Deformation of a Complete System**

3.1. Introduction

Elementary, separate objects display simple mechanisms of deformation, but when multiple complex objects with different characteristics are deformed together, the result can be quite complicated to calculate. The real world system chosen for this project is vehicular deformation. Vehicular deformation is quite spectacular, as different body panels of the car get bent, small parts fall off and windows shatter. This section outlines the structural design of a standard vehicle and the factors which affect vehicular deformation.

3.2. Structural Design and Collision of Automobiles

Vehicles come in all shapes and sizes, but they all share similar structural components, such as the chassis. The chassis is the main supporting structure of a vehicle. It acts as the endoskeleton of a vehicle, as it protects the components and occupants from external forces. The most common chassis type used in commercial cars nowadays is the unibody frame, where the body, floor and chassis form a single structure [4]. A unibody chassis is shown below in Figure 3.1. The chassis should stay as rigid as possible to prevent injury to the inhabitants during a collision. Therefore, modern unibody frames are typically constructed from carbon steel [6], which is less ductile and more brittle than normal steel [7]. Upon crashing, the frame arms extending from the central monocoque are designed to undergo controlled deformation in the crumple zone, which safely redistributes the forces exerted on the car during a collision [8].

Figure 3.1: A modern unibody chassis. [5]

The vehicle demonstrated in this project will utilise a front-engine layout, which means that frontal collisions to the car should display less deformation than rear collisions. Modern cars come equipped with an breakaway engine mount, which is a safety feature which allows the engine to separate from the chassis during high speed frontal collisions [9]. This prevents the solid engine block from breaking into the monocoque, therefore keeping the front occupants safe. Slow collisions do not trigger the separation of the mount, which means that the front of the car would show high deformation up to the bounds of the engine block, and then minimal deformation as the engine block would absorb the forces of the collision. The engine block is typically made from a solid block of aluminium alloy [10], which is very difficult to deform under the compressive stress of a collision. The more fragile parts of the engine assembly such as the battery, fluid containers, and radiator would easily get deformed and separated from the vehicle during collisions.

Body panels form the remainder of the structural components of a vehicle. These can include bonnets, bumpers, doors and fenders. These components are constructed from long, relatively thin panels of aluminium, which is a malleable metal that can be easily deformed during collisions [11]. Body panels such as fenders are attached to the chassis using several bolts. The areas of the panel in proximity to these connection points will deform much less than the central parts of the panel, which will likely buckle and bend under the compressive forces of a collision. This can be seen on the door panel in Figure 3.2. Moveable body panels such as doors are attached to the chassis using hinge joints, which can easily break during collisions.

Figure 3.2: Crash test of a Volkswagen car. [12]

Chapter 4. **Technologies**

4.1. Introduction

It is important to choose the ideal set of technologies for use in this project, since the use of unsuitable technologies will have a profound negative impact on the project by slowing the development velocity and reducing the overall quality of the project. This section compares the physics engines and 3D modelling software examined for use in this project.

4.2. Physics Engine

Physics engines are computer software which provide an approximate simulation of physical systems. Soft body dynamics is not natively supported in some physics engines, so many physics engines were considered for use in this project. This section outlines the different physics engines which were considered, their features, their disadvantages, and their support for soft body simulation.

The first physics engine considered was Unreal Engine 4, developed by Epic Games. UE4 contains a complete suite of development tools designed for real time technology. UE4 is written on C++, a cross-platformed language that can be used to create complex, high performance applications. UE4 has many features, including blueprints, post-processing, advanced artificial intelligence, and a marketplace ecosystem [13]. However, UE4 is much more suited to long term projects such as triple-A games, rather than small projects such as this final year project. UE4 does not natively

support soft body simulation, but plugins can be downloaded from the marketplace which can perform this task.

The second physics engine considered was Lumberyard, developed by Amazon. Lumberyard is based on Crytek's CryEngine, a proven engine used for video games such as the Far Cry series. Lumberyard is written on C++ and Lua, a lightweight, high-level, multi-paradigm programming language designed for embedded use. Lumberyard has many features such as integration with Amazon Web Services for multiplayer and streaming features, a convenient real time lighting engine, and incredible graphics capabilities [14]. The biggest drawback of Lumberyard is that this engine is still in beta testing, so the userbase is small and the support is minimal. Lumberyard does not natively support soft body simulation.

The third physics engine considered was Unity, developed by Unity Technologies. Unity is a popular cross-platform games engine which supports both 2D and 3D development. Unity is written in C#, a highly popular language that is simple, modern, and general purpose. Unity has many features, including customisable editors, post-processing, artificial intelligence pathfinding, and a marketplace ecosystem with a vibrant community [15]. However, Unity has certain drawbacks such as being closed-source and a subpar graphics pipeline. Unity does not natively support soft body simulation.

Unity was chosen as the physics engine for this project due to its ease of development and support.

4.3. 3D Modelling Software

3D modelling software is computer software which aids the user in creating a mathematical representation of a 3D object. In this project's context, 3D modelling will be used to create game object meshes which have a clean and consistent topology, which is crucial to realistically model soft body dynamics. Complex 3D modelling is not natively supported in most physics engines, including Unity, and therefore a separate 3DMS is required. Only two 3DMS were considered since other mainstream 3DMS were too expensive for use in this project. This section outlines their features and their disadvantages.

The first 3DMS considered was Autodesk Maya, developed by Autodesk. Maya is the industry standard for computer graphics, and is used in many companies around the world. Maya contains an incredible amount of features, such as hair, cloth, and particle simulation [16], but it is a very complex software that takes long to master. Due to the small timeframe for this project, Maya was not chosen for use.

The second 3DMS considered was Blender, developed by Blender Foundation. Blender is hugely popular in industry and hobbyists alike, due to its ease of development and open-sourced nature. Blender contains many features such as sculpting, decimation of polygons, remeshing, all of which are ideal for use in this project [17].

Blender was chosen as the 3DMS due to its ease of development, numerous features, and the ability to import Blender models directly into the Unity physics engine.

Chapter 5. Soft Body Physics Implementation

5.1. Introduction

Unity does not natively support soft body simulation for vehicular collisions, but using the research of deformation physics and the structural properties of modern vehicles, a simple yet effective implementation can be created from scratch in the Unity physics engine. This section outlines a simple approach from creating a model in Blender to implementing soft body simulation in Unity.

5.2. Creating Models in Blender

Unity does not natively support 3D modelling. Some primitive 3D modelling extensions are available in the asset store, but these extensions fall well short of the functionalities which standalone 3DMS packages have to offer. Creating models in Blender is an easy process, and some models have already been created using Blender for this project.

Once the technical specifications of an object has been found/decided, the object is modelled in Blender. Tools such as Boolean (used to cut objects into objects), Subdivide (used to increase the number of polygons in an object, so more accurate edits to models can be made) and Bevel (used to smooth corners) can be used to create an accurate 3D representation of the physical object in mind. Afterwards, the topology (the characteristics of the polygons on a surface) of the model needs to be finalised. This is an extremely important step, since the vertices on the model's mesh dictate the nature and realism of the soft body collision. The topology can be conveniently edited using the Re-mesh tool, as shown in Figure 5.1. While one should strive to increase the complexity of the model's mesh, it is very important to take the polygon budget into consideration. An increase in the number of polygons will bring about more detailed meshes, and therefore more realistic soft body interactions, but the application will also require more resources to compute the position and velocity of each vertex. Therefore, a compromise needs to be made between the realism and performance of the simulation. The balance can only be found through experimentation.

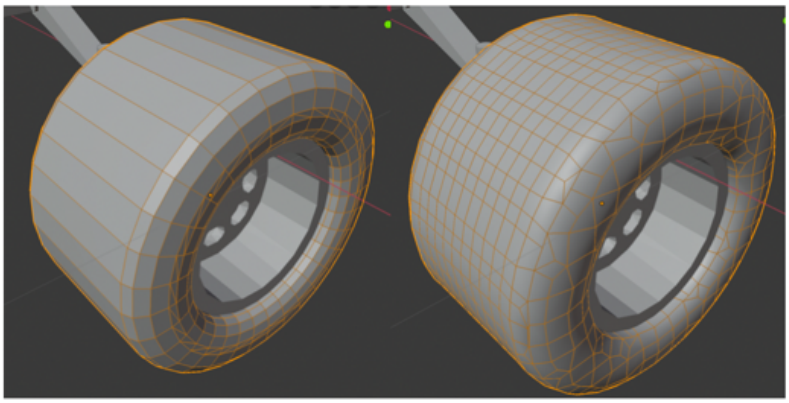

Figure 5.1: The tyre on the left has unsuitable topology. The tyre on the right now has suitable topology after re-meshing.

Once the model has been created, colours and materials can be applied to the object using Blender's shading tools. Alternatively, textures can be applied at a simpler level in Unity's inspector tab. Afterwards, the blender file can be directly imported into Unity, where it can be added to the scene.

5.3. Implementation of Soft Body Physics in Unity

In computer graphics, the model is portrayed as a collection of vertices which are connected by edges. The areas bounded by the edges are displayed as faces. The position of individual vertices can be manipulated to change the shape of the object, hence enabling soft body simulation. To perform any deformation in Unity, we need to access the GameObject's mesh. The mesh stores many useful parameters, some of which are shown in Table 5.1.

Property	Description
Vertices	An array of every vertex in the mesh, stored as Vector3.
UV	An array of 2D coordinates which can map textures onto 3D objects
Triangles	An array of triangles, whose position is determined by its indices in the vertices array.
Normals	An array of normals, which are Vector3 vectors pointing outwards perpendicularly from vertex. Used for calculating shading.

Table 5.1: The mesh properties to be manipulated for soft body simulation.

A soft body interaction takes place when a GameObject with soft body dynamics interacts with other GameObjects in the scene. The interaction can be analysed using colliders, which are layers on a GameObject which can detect collisions. Primitive colliders such as box colliders and sphere colliders can approximate the shape of a GameObject while keeping a low resource overhead, but mesh colliders can match the exact shape of the mesh exactly at a cost of high processor overhead. Real time mesh collider manipulation significantly degrades performance, since a moving body near a mesh collider checks for collisions against every triangle in the mesh. Even though using mesh colliders is the absolute ideal approach to detecting soft body collisions, a compromise has to be made to limit performance loss.

Possible solutions to this technical challenge will need to be explored over the course of the project. Some solutions already exist, such as using compound colliders, which is a combination of primitive colliders that can offer more accurate collision detection than a single primitive collider. Another potential solution is to use bounding volume hierarchies to approximate colliders, where polygons are recursively nested into groups of box colliders, with each group having larger volume than the group it contains [18], as shown in Figure 5.2. This approach has been used by some video games to implement ray tracing, but it has not been used with collision detection before, and may be an interesting approach to maintain simulation realism and performance.

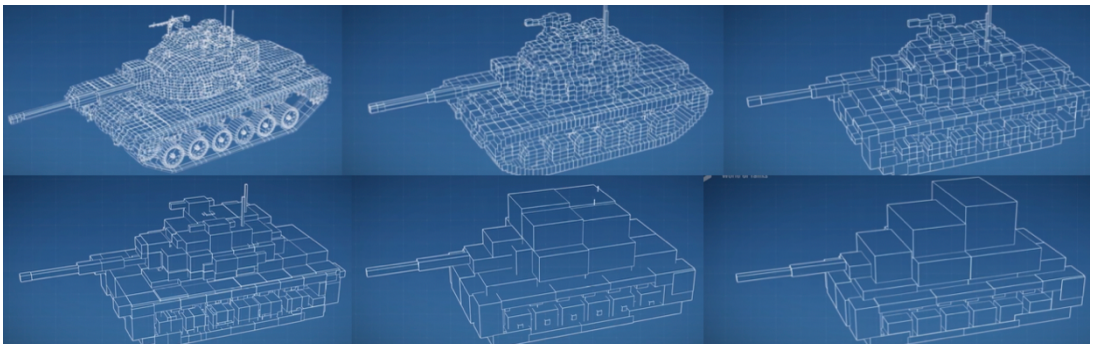

Figure 5.2: The BVH shown with progressively larger volumes and polygons contained. [18]

Deforming forces do not act in a single direction, instead the forces are applied in all directions equally from the contact point. In Unity, the force should not be applied directly onto the surface of the contact, since it will give unwanted results. Instead, it is appropriate to offset the force from the surface of the contact point. To do this, the normal of the contact point can be multiplied by the desired offset. As the mesh is deformed, its vertices have velocities due to the movement. Vertices that are further away from the contact point experience less deformation that vertices closer to the contact point. The force exerted on each vertex can be calculated by modifying the Inverse-Square law, as shown in Figure 5.3. Adjusting values of *x* defines the strength of the mesh at that vertex, and adjusting the coefficient of d^2 defines the rate of force degradation from the contact point. Using this formula, the structural characteristics of any object can be simulated.

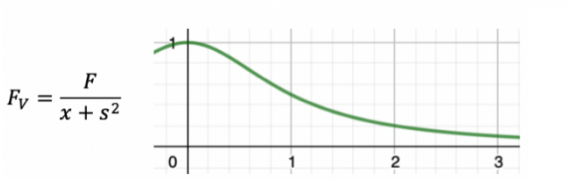

Figure 5.3: A modified Inverse-Square Law

After calculating the force at the vertex, the acceleration of the vertex can be found using Newton's second law of motion, $F = ma$. After calculating the acceleration, the velocity of the vertex can be found using Newton's laws of motion, $v = u + at$. This velocity can be applied to the vertex along the vector which points from the contact point to the vertex, which will model the deformation over time. In reality, the velocities of the displaced vertices are reduced over time, since an elastic force acts in the opposite direction of the deformation force. This can be modelled using Hooke's law, $F = -ks$, since this is just a simple spring system. Using Newton's second law of motion and laws of motion, the velocity of the vertex in the opposite direction of the deformation can be calculated as $v = -kst$. Once this velocity is added to the velocity of the vertex from the deformation alone, the actual resultant velocity of the vertex can be obtained. As the vertex is displaced further from its original resting position, the elastic force acts stronger, hence reducing the resultant velocity of the vertex.

For plastic deformation, the deformation ceases once the resultant velocity is zero. For elastic deformation, the vertices need to return to their original point. If spring damping was not present, the vertex will move between its resting position and maximum displacement position infinitely. To add a damping factor to the spring system, the final resultant velocity is multiplied by a negative damping factor, which ensures that the vertex will eventually return to its original resting position. Upon each frame, the normals of the mesh will need to be recalculated to change the shading appropriately to match the deformation.

References

[1] Lumen (2019), *Elasticity, Stress, Strain, and Fracture*, viewed 20 October 2019 <https://courses.lumenlearning.com/boundless-physics/chapter/elasticity-stress-strain-and-fracture/>

[2] NDT Resource Centre (2019), *Elastic/Plastic Deformation*, viewed 20 October 2019 <https://www.nde-ed.org/EducationResources/CommunityCollege/Materials/Structure/deformation.htm>

[3] Ashish (2019), *Tensile strength of concrete*, viewed 20 October 2019 <https://www.scienceabc.com/pure-sciences/why-does-concrete-have-great-compressive-strength-but-poor-tensile-strength.html>

[4] Benjamin Jerew (2018), *A … B … CAR CHASSIS … KNOW YOUR CAR PARTS*, viewed 21 October 2019 <http://knowhow.napaonline.com/a-b-car-chassis-know-your-car-parts/>

[5] EmmyCN (2019), *The two main types of vehicle chassis and their qualities*, viewed 21 October 2019 <https://mobilityarena.com/two-main-types-vehicle-chassis-qualities/>

[6] Hubert Vester Honda (2018), *1. What Is a Chassis?*, viewed 21 October 2019 <https://www.hubertvesterhonda.com/4-chassis-details-know/>

[7] Admin (2015), *What is Carbon Steel*, viewed 21 October 2019 <https://pediaa.com/difference-between-carbon-steel-and-mild-steel/>

[8] Ed Grabianowski (Unknown), *How Crumple Zones Work*, viewed 22 October 2019 <https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/crumple-zone.htm>

[9] Shaun Patrick Davidson (2008), *12 Important Safety Features On Our Cars We Don't Think About*, viewed 23 October 2019 <https://ezinearticles.com/?12-Important-Safety-Features-On-Our-Cars-We-Dont-Think-About&id=1073393>

[10] Nancy Lovering (Unknown), *How Is an Engine Block Made?*, viewed 23 October 2019 <https://itstillruns.com/engine-block-made-7606610.html>

[11] Adam Hornbacher (Unknown), *Strength & Malleability of Steel vs Aluminium*, viewed 23 October 2019 <https://www.wenzelmetalspinning.com/steel-vs-aluminum.html>

[12] Cheryl Jensen (2012), *Tougher Crash Test Brings Lower Scores*, viewed 23 October 2019 <https://www.nytimes.com/2012/08/19/automobiles/tougher-crash-test-brings-lower-scores.html>

[13] Epic Games (2019), *Features*, viewed 25 October 2019 <https://www.unrealengine.com/en-US/features>

[14] Amazon (2019), *Amazon Lumberyard*, viewed 25 October 2019 <https://aws.amazon.com/lumberyard/>

[15] Unity Technologies (2019), *Explore Unity*, viewed 25 October 2019 <https://unity3d.com/unity?_ga=2.142127482.977862438.1572809257-1157649060.1572558565>

[16] Autodesk (2019), *Features*, viewed 25 October 2019 <https://www.autodesk.eu/products/maya/features>

[17] Blender Foundation (2019), *Features*, viewed 25 October 2019 <https://www.blender.org/features/>

[18] World of Tanks Europe (2019), *Features*, viewed 29 October 2019 <https://www.youtube.com/watch?v=dXbjmF--QVc>