

Computer Games Development

Project Report

Year IV

**Mantas Zalnierius**

**C00241626**

**28/10/2021**

# Declaration

# This examination/assessment will be submitted using Turnitin as the online submission tool. By submitting my examination/assessment to Turnitin, I am declaring that this examination/assessment is my own work. I understand that I may be required to orally defend any of my answers, to the lecturer, at a given time after the examination/assessment has been completed, as outlined in the student regulations.

**Contents**

[Acknowledgements 2](#_Toc54714373)

[Project Abstract 2](#_Toc54714374)

[Project Introduction and/or Research Question 2](#_Toc54714375)

[Literature Review 2](#_Toc54714376)

[Evaluation and Discussion 3](#_Toc54714377)

[Conclusions 3](#_Toc54714378)

[References 4](#_Toc54714379)

[Appendices 4](#_Toc54714380)

# Acknowledgements

# Project Abstract

This project is to make a 2D platformer game with a custom physics system using robust collision detection in SFML.

The goal of the project is to implement a 2D physics and collision library that anybody can use to make their own 2D platformer games. For the physics aspect of my project, I will use ADSR envelopes to have an immersive symmetrical movement. For the 2D collision library I would use algorithms such as AABB (Axis-Aligned Bounding Box), SAT (Separating Axis Theorem) and Diagonals algorithms. I will also incorporate techniques that will make these expensive mathematical operations fast and responsive. I will also use multiple programming patterns. So that my project will have a very standardized codebase that anybody will be able to understand and use.

At the end of the project what I will expect to have done, is fully done game containing 2-3 Levels that use all the Collisions and physics that I have mentioned to have a game that feels immersive when playing the game.

# Project Introduction and/or Research Question

In a traditional 2D game platformer game that have a lot of entities consisting of different types of shapes (convex & concave). All these entities need to have collision detection and physics related movement. Collision detection by itself is hard to implement in a manner that results in maximum efficiency for users’ computers and still look immersive. This type of efficiency comes from different algorithms (or in some cases combined algorithms). There’s also physic based movement, having a system that has movement that is fixed is very not immersive to the user. Having a system that makes the game feel responsive and immersive takes a lot of time to as the developer and designers need to know how far the player can jump in their game that feels right to the user. This is very important in the development of 2D platformer games because if the collision detection feels not immersive (in this case where the collision detection displacement causes scenarios where it doesn't look right) or if people had less powerful machines it could cause slowdown that makes the game unplayable to the user.

In this paper will look at a multitude of different ways of doing collision in a 2D game. One of the most used algorithms is AABB. The reason for this is because it’s rather easy to implement and not many maths operations are used in AABB. The problem with AABB collision is that it cannot handle collisions between rectangles that are rotated which is a big draw back. That’s where SAT comes in. This algorithm is able to correctly handle collision detection between rotated rectangles. The issues with SAT are that it’s a lot harder to implement, but the implementation will cause 2D game to feel more immersive.

In respect to physics, I will be using an ADSR (Attack, Decay, Sustain, Release) envelope, where each letter will stand for stages in movement for a character in a video game. In the attack stage the character needs to overcome static friction to actually move. The Decay stage is where the character is slowed down due to kinetic friction. Then on the Sustain stage the character will be able to move for sometime until he gets to the relapse stage where the character will slow down to a halt. This leads to very symmetrical movement that feels immersive to the player. With collision detection and physics-based movement implemented, it will lead to a game that feels immersive.

In this paper, I show how good collision detection and movement will lead to a game that will make user feel immersed while playing the game without any slow down on their personal machines. The process that I will take to implement the collision detection will be the use of the SAT algorithm. For the physics-based movement I will create an ADSR algorithm to make immersive symmetrical movement.

Questions to answer

* How can collision/Physics be managed when there are hundreds of objects in the game at once?
* How will my collisions/Physics perform in environments where the users Personal Computer has a dedicated 8 Gb’s of Ram and have a processer that has 2.0 GHz or greater speed.
* Are there better Collision/Physics algorithms that do the same thing but at a faster computational time (Algorithms such as the Diagonals vs SAT)?

# Literature Review

“**SAT (Separating-Axis- theorem)**

*Separating axis theorem by S. Gottschalk [2] suggested that the main content is for two or more objects will collide, if I could find an axis so that these objects do not overlap on the projection of the axis, then we can believe that these objects do not intersect each other. Select theoretically axis is not fixed, as long as there is such an axis. However, due to the separation axis theorem applies to convex polygons, so for non-convex polygon can be decomposed into several all convex polygons.*

*For 2D, the normal of each edge of convex polygon include the possibility of the axis in all directions, so when we select potential separating axis (Potential Separating Axis, PSA), generally we can choose normal of edges of the convex polygon.*

*For negative numbers, it just represents different directions of the two normal vectors, and it does not need to be considered when vertexes projected onto potential separating axis. Separating axis theorem collision detection algorithm is an optimistic collision detection algorithm, once you have found such a disjoint axis that detection will no longer be implemented, otherwise would have been implemented. And when the convex polygon becomes more complex, the greater the corresponding overhead. So, for these shortcomings, leading to the emergence of a large number* of *calculations, there are many optimization schemes. The main idea is to reduce the number of potential separating axis, thereby reducing the amount of computation for collision detection. For example, two rectangles, because of its edges are parallel two by two, so we can merge potential separating axis into two separating axis, Under the best of circumstances the number of its maximum potential separating axis is two. The worst case is four. There is no need to test eight axes, we just need to detect two or four axes to determine whether the collision. Unfortunately, other irregularities cannot use this method, only to detect potential separating axis one by one. There are other optimization programs, such as the object's bounding box beforehand rough test, then more precise AABB bounding box test [12], after separating axis test, thereby reducing unnecessary calculations. Clearly these measures did not solve this question from within.* *“*

<http://www.ijscience.org/download/IJS-2-10-110-114.pdf>

# In this article in details what SAT is and how it can get around having different types of convex and non-convex shapes and how it can handle collision even if the shapes are rotated.

This is done by projecting all the points on every convex or non-convex shapes normal. If the projections overlap between 2 shapes normal, if it does that means there is collision between those shapes. With SAT, it has a rather large computational time, has you need to calculate every shape projection over another and see if they are overlapping, which will lead to slow down on some machines. To fix this, I will implement a spatial partition, where the 2D world will be laid out in a grid. This will mean that I will only need to check for collisions only if two or more shapes are in a cell in that grid, in turn reducing the time complexity of SAT.

“**ADSR Envelope**

*ADSR stands for “Attack, Decay, Sustain, Release”, which are the four phases of inputs, both digital and physical. Swink uses the example of plucking a guitar string to illustrate: The Attack comes from initially hitting the string, the Decay from the drop in initial volume, the Sustain from holding the note, and the Release from muting the string or letting go of a fret. Believe it or not, this directly correlates to Newtonian physics: an object needs to overcome static friction before it can start moving (Attack), and is then slowed by a smaller kinetic friction value (Decay). The object then moves at a fairly constant speed (Sustain) before being slowed completely by friction or other forces (Release). If an object at least looks like it obeys physics, we will interpret it as doing so.”*

In this book it details how an ADSR envelope can be used in games to create physics-based movement that have weight to them, which in turn creates a game that feels immersive to the user. With an ADSR envelope it uses 4 different states, Attack which is the initial state after an action such as moving/jumping. Then after the Decay state it will enter the Decay state where physics such as a small drag/friction force will be applied that will slow down the object. After the Decay state it will enter the Sustain state, this state is where the object will move at constant speed for a time. Until it enters the Release state where the object will be slowed to a complete stop by strong physics forces such as drag/friction. Using ADSR will lead to movement that feels symmetrical. (I.e., if you jump into the air, it will take the same amount of time to reach the max height, as it is falling to the ground.)

<https://jaredemitchell.com/senior-capstone-blogs/proof-of-concept-applying-game-feel-113015/>

# Evaluation and Discussion

**Project Milestones**

**Milestone #1**

Build a game in SFML with a game loop working (28/10/2021)

* Set up game using SFML\_SDK environment variable.
* Have assets folder set up containing fonts images etc.

**Milestone #2**

Created first draft of project research document (11/11/2021)

* First draft of research document submitted.

**Major Technical Achievements**

**Project Review**

# Conclusions

**Future Work**

# References

# Appendices