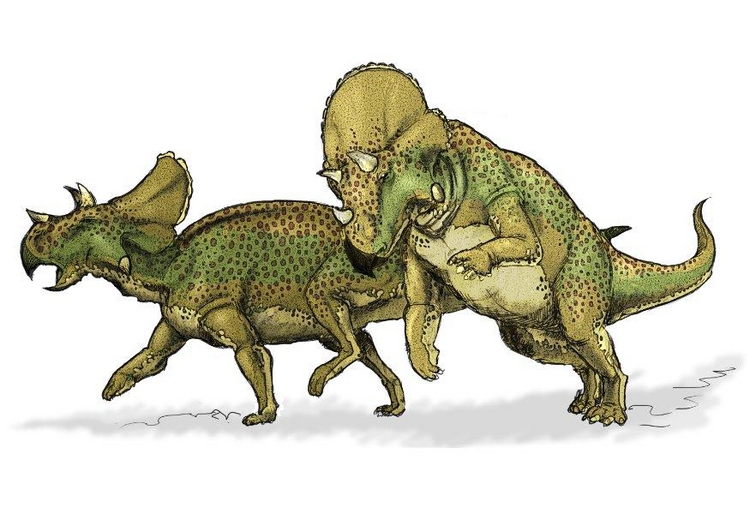
4/26/2010

|  |
| --- |
| Development of software modules for physics and AI engines for computer games | Alex J Davison (DV003874) |



Dinosaur (2010)

|  |  |
| --- | --- |
| CE00851-2 | Programming Physics and AI Engines for Games |

Contents

[Introduction 1](#_Toc259609571)

[Algorithms or techniques used 1](#_Toc259609572)

[Movement of game objects (Constant acceleration and velocity) 1](#_Toc259609573)

[Dinosaur jumping (Projectile motion) 2](#_Toc259609574)

[Asteroids bouncing/exploding and dinosaurs continuing jumping (Collision detection and response) 2](#_Toc259609575)

[Explosions (Practical system for special effects) 2](#_Toc259609576)

[Swarm Movement (AI Component/Practical Emitter) 3](#_Toc259609577)

[Critical appraisal 3](#_Toc259609578)

[Conclusion 3](#_Toc259609579)

[Bibliography 4](#_Toc259609580)

[Appendix 4](#_Toc259609581)

[User guide 4](#_Toc259609582)

[Files stored on the disk 4](#_Toc259609583)

[Screen shots 4](#_Toc259609584)

[Architecture block-diagram 5](#_Toc259609585)

# Introduction

This report accompanies a program which shows how physics and AI (Artificial Intelligence) have been implemented within a graphical application. This report contains an architecture block diagram that shows the structure of the application, a description of the algorithms and/or techniques used within the application to achieve believable physic and AI. This report also covers a critical appraisal of how well the application has been implemented and any difficulties faced using the implementation. The report will also provides a brief user guide, CD content table and a screen shot to show how the physics and AI have been implemented with the graphical application.

# Algorithms or techniques used

Below is a breakdown of the algorithms and/or techniques used within the application:

## Movement of game objects (Constant acceleration and velocity)

Each game object has a particle object which allows the game object to move on the x and y axis of the 2D world. Each particle has a force applied to it and the particle has a mass. Using the formula below the acceleration of the particle can be calculated.

The displacement is then calculated by adding the old position to the newly created position. The velocity and time are used to find the new position, using the following formula.

Once the acceleration and displacement has been calculated the velocity is then updated using the following formula.

This movement technique was implemented rather than others, available because less data is required. Other techniques require data on initial velocity, initial position etc which creates larger figures to work with as well as the increase in storage overhead for the extra data required.

## [[1]](#footnote-5)Dinosaur jumping (Projectile motion)

The dinosaurs that jump along the bottom of the screen at an angle use the formulae set out below to work out the velocity for the x and y axis. This calculation is carried out only once. Then the movement of game object calculations is carried out. Below are the calculations used to calculate the velocity of the x and y axes.

## [[2]](#footnote-6)Asteroids bouncing/exploding and dinosaurs continuing jumping (Collision detection and response)

For each asteroid game object there is an equal particle system object. The asteroids moving in the world use collision detection to ensure they move within the confines of the world only. The collision detection for axes aligned object is a simple greater than or less than check of the particle of the game object. However the collision detection for the non axis aligned collision detection uses the following technique and formulas.

The slope of the asteroids velocity is calculated, as well as the slope of the plane in which the asteroids are likely to hit. Below is the calculation to find the two slopes. The two y and two x elements make up the two points on the line used to work out the slope of the line.

Once the two slopes have been calculated then the two points on the two lines (one from the asteroid and one from the plane itself) are used to work out the point at which the two points intersect one another. Below is the calculation used to work out the point at which the two lines intersect on the x axis.

Below is the calculation used to calculate the point at which the two lines intersect on the y axis.

If the collision occurs on the edges of the world viewing port or the cover for the dinosaur’s layer then the asteroids seemingly bounce off the edges at which the collision occurred. The bouncing of the asteroids was achieved by doing the following calculation once the collision had occurred. Below is the x and y calculation.

These calculations give the impression of an elastic collision with no energy loss from sound and/or heat. To improve the calculation a random number between one and zero could have been used to reduce the velocity of the asteroid after the collision giving the impression of energy loss from sound and/or heat.

## [[3]](#footnote-8)Explosions (Practical system for special effects)

The explosions that occur, once the collision of the asteroids with the bottom of the world has been detected, are achieved as follows. The point at which the collision occurs is the point at which all the particles for the explosion are set. Then the particles are given a random force in the x and y axes and a random mass, which allows them to look random. Then the particles move out from the point from which they were set to.

Once the particles have started moving a check is carried out to ensure that the particles do not move more than one hundred pixels away from the origin of the explosion. Below is the calculation used to work out the magnitude of the particles from the origin. The position of the origin of the explosion is subtracted from the position of the particles within the world. Once the particle is greater than one hundred pixels away from the origin then the piratical is no longer drawn.

## Swarm Movement (AI Component/Practical Emitter)

The swarm movement and direction are determined by a finite state machine which determines the state at which the swarm moves. Although the individual flies (particles) do not have a finite state machine, the swarm as a whole does. A particle as acts as the main fly and all the other flies follow. The flies move randomly and if one of the flies’ moves out of range of the main fly, then the rogue fly is simply redraw at the main fly’s position. Determining if the fly is a rogue fly is a simple case of determining the magnitude/distance between the main fly and all the other flies.

It would have been possible to implement if the main fly, flies within visible distance of a dinosaur then the swarm finite state machine is change to make the swarm move towards the dinosaur. However, if the main fly cannot see any dinosaur then the swarm continues to patrol around the world with the predefine points plotted.

# Critical appraisal

The functionality and performance of the application is very good however there are some improvements that could have been made. More physic could have been used to create a far better realistic world, however, this would have increased the processing time need to draw a frame.

The implementation of gravity is efficient as it is added to the acceleration, so the gravity for each object does not have to be calculated separately. Also the response of the elastic collisions is easily calculated as the x and y velocities are just inversed rather than manually calculated using the calculation below.

The x and y velocity are multiplied by negative one to reverse the sign of the velocity of the x and y component of the velocity.

The storage of game objects has a low overhead due to the structure of the game objects and the structure of the particle object within the game object. The primitives and the structs used within the structures of the game object take up miniscule amounts of memory on current computers. This allows for many game objects/particles to be declared and used within the world. During testing of the particle system, up to ten thousand particles were used in one given scenario to give the impression of snow/rain, however, this did not meet the assignment scenario outline and was removed at a later date. This should the lower overhead of storage required and the efficient calculations used.

# Conclusion

The application fulfils many of the requirements that are outlined in the assignment. The application carries out the following requirements outlined in the assignment brief, “...(i) rectilinear motion at constant velocity or acceleration...”, “...(iv) projectile motion which includes the effect of gravity...”, “...(v) collision response...” and “(vi) particle system for explosion or other special effects...” Dr. Claude Chibelushi (2010). Overall the assignment and application were a great success and a great learning experience. The execution of more physic would have increased the believability of the application had more time been available. The application achieves the outlined items well as the application looks believable and carries out many of the physics calculations covered in the lectures Overall a good learning experience that was well executed.

# Bibliography

|  |
| --- |
| Dr. Claude Chibelushi (2010). Development of software modules for physics and AI engines for computer games p. 2. |
| Dinosaur (2010) Edupics [Online]. Available from: http://www.edupics.com/avaceratops-dinosaur-t7602.jpg [Accessed: 16 April 2010] |

# Appendix

## User guide

|  |  |
| --- | --- |
| Key | **Response** |
| Left Arrow | Changes the direction of the panel/box under the UFO so that it moves left rather than right. |
| Right Arrow | Changes the direction of the panel/box under the UFO so that it moves right rather than left. |
| Esc | Ends Program |

## Files stored on the disk

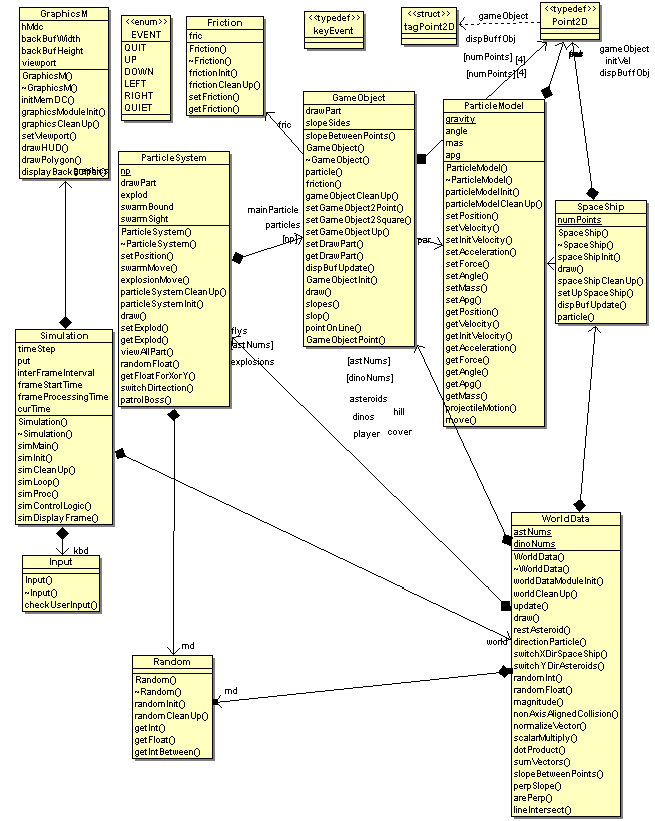
|  |  |
| --- | --- |
| Files | **Description** |
| PPAIE4GProC++\_GDIR1 | Folder containing the entire visual studio 2008 project. |
| Assignment.xdoc | Copy of the assignments support documentation.[[4]](#footnote-9) |
| UML | Folder containing a BOUML created UML |

## Screen shots

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

## Architecture block-diagram

Below is a block-diagram depicting the architecture of the software:



1. One of the best physics or AI software components implemented. [↑](#footnote-ref-5)
2. One of the best physics or AI software components implemented. [↑](#footnote-ref-6)
3. One of the best physics or AI software components implemented. [↑](#footnote-ref-8)
4. This document. [↑](#footnote-ref-9)