Bibliography

Books:

- AQA (2006) *GCE Mathematics Further Pure unit 4 Textbook*. Available from: http://filestore.aqa.org.uk/subjects/AQA-MFP4-TEXTBOOK.PDF [Accessed: 22nd September 2013]
- Goldman, R. (1990) Intersection of Two Lines in Three-Space. In: Glassner, A. (ed.) *Graphics Gems*. Academic Press Inc., p. 304.
- Woo, M., Neider, J., Davis, T., and OpenGL Architecture Review Board (1997) *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.1*. 2nd edition. Addison-Wesley Publishing.

Papers:

- Cline, D., Cardon, D., and Egbert, P. K. (2007) Fluid Flow for the Rest of Us: Tutorial of the Marker and Cell Method in Computer Graphics. Available from: http://people.sc.fsu.edu/~jburkardt/pdf/fluid_flow_for_the_rest_of_us.pdf [Accessed: 22nd September 2013]
- Green, C., (2007) Improved Alpha-Tested Magnification for Vector Textures and Special Effects. In: *Proceedings of ACM SIGGRAPH 2007*. Available from: http://www.valvesoftware.com/publications/2007/SIGGRAPH2007_AlphaTestedMagnification.pdf [Accessed: 22nd September 2013]
- Jakobsen, T. (2001) Advanced Character Physics. In: *Proceedings of the Game Developers Conference 2001*. Available from: http://web.archive.org/web/20080410171619/http://www.teknikus.dk/tj/gdc2001.htm [Accessed: 22nd September 2013]
- Jiarathanakul, P. (2012) Ray Marching Distance Fields in Real-time on WebGL. University of Pennsylvania. Available from: http://www.seas.upenn.edu/~pjia/raymarch/report.pdf [Accessed: 22nd September 2013]

Lectures & Presentations:

- Quilezles, I. (2008) *Rendering Worlds with Two Triangles with raytracing on the GPU in 4096 bytes* [Presentation] NVScene 2008. Slides available from: http://www.iquilezles.org/www/material/nvscene2008/rwwtt.pdf [Accessed: 22nd September 2013]
- Swoboda, M. (2012) *Advanced Procedural Rendering in DirectX 11* [Presentation] Game Developers Conference 2012. Slides available from: http://directtovideo.files.wordpress.com/2012/03/gdc_2012_released.pdf [Accessed: 22nd September 2013]
- Zorin, D. (2004) *Mesh data structures*. [Lecture] Interactive Graphics course, Fall 2004. New York University. Slides available from: http://mrl.nyu.edu/~dzorin/ig04/lecture24/meshes.pdf [Accessed: 23rd September 2013]

Websites:

- Astle, D. (2006) OpenGL Frame Buffer Object 101. Available from: http://www.gamedev.net/page/resources/_/technical/opengl/opengl-frame-buffer-object-101-r2331 [Accessed: 23rd September 2013]
- Atkins, J. (2013) SDL_ttf documentation. Available from: http://www.libsdl.org/projects/SDL_ttf/docs/SDL_ttf.html [Accessed: 23rd September 2013]
- Bock, A. (2011) SDL TTF: Fonts, and how to use them. Available from: http://www.sdltutorials.com/sdl-ttf [Accessed: 23rd September 2013]

- Boesch, F. (2010) Hard Constraints, Easy Solutions. *Codeflow*. Weblog. Available from: http://codeflow.org/entries/2010/sep/01/hard-constraints-easy-solutions/ [Accessed: 23th September 2013]
- CFD Online (2012) Navier-Stokes equations. Available from: http://www.cfd-online.com/Wiki/Navier-Stokes_equations [Accessed: 23rd September 2013]
- confuted (2004) Using Quaternion to Perform 3D rotations. Available from: http://www.cprogramming.com/tutorial/3d/quaternions.html [Accessed: 23rd September 2013]
- Côté, A. S., Smith, W., and Lindan, P. J. (2001) The Basic Verlet Algorithm. *Democritus: A Molecular Dynamics tutorial*. CCLRC Daresbury Laboratory. Available from: http://www.compsoc.man.ac.uk/~lucky/Democritus/Theory/verlet.html#verlet [Accessed: 23rd September 2013]
- Giesen, F. (2012) Half-edge based mesh representations: practice. *The ryg blog*. Weblog. Available from: http://fgiesen.wordpress.com/2012/03/24/half-edge-based-mesh-representations-practice/ [Accessed: 23rd September 2013]
- Giesen, F. (2012) Half Edges Redux. *The ryg blog*. Weblog. Available from: http://fgiesen.wordpress.com/2012/04/03/half-edges-redux/ [Accessed: 23rd September 2013]
- Hoetzlein, R. C. (2010) Surface Reconstruction of SPH Fluids. Weblog. Available from: http://www.rchoetzlein.com/theory/2010/surface-reconstruction-of-sph-fluids/ [Accessed: 23rd September 2013]
- Kammerl, J. (2011) Spatial Partitioning and Search Operations with Octrees. Available from: http://pointclouds.org/documentation/tutorials/octree.php [Accessed: 23rd September 2013]
- Khronos Group et al. (2013) OpenGL Core API Reference. Available from: http://www.opengl.org/wiki/Category:Core_API_Reference [Accessed: 23rd September 2013]
- mandarine (2013) pouët.net. Available from: http://www.pouet.net [Accessed: 3rd October 2013]
- McKesson, J. L. (2012) Learning Modern 3D Graphics Programming. Available from: http://www.arcsynthesis.org/gltut/ [Accessed 23rd September 2013]
- navis (2010) 2007 and now. *Iconoclash*. Weblog. Available from: http://navis-asd.blogspot.co.uk/2010/04/2007-and-now.html [Accessed: 23rd September 2013]
- OpenSceneGraph (2012) What is a scene graph? Available from: http://www.openscenegraph.org/index.php/documentation/knowledge-base/36-what-is-a-scene-graph [Accessed: 23rd September 2013]
- Stack Overflow. (2011) OpenGL - glm::perspective explanation. Available from: http://stackoverflow.com/questions/8115352/glmperspective-explanation [Accessed: 23rd September 2013]
- Swoboda, M. (2011) Numb res. *Direct to video*. Weblog. Available from: http://directtovideo.wordpress.com/2011/05/03/numb-res/ [Accessed: 23rd September 2013]
- Willemse, R. (2000) The Water Effect Explained. Available from: http://www.gamedev.net/page/resources/_/technical/graphics-programming-and-theory/the-water-effect-explained-r915 [Accessed: 23rd September 2013]

Software and code:

- Falstad, P. (2009) *Ripple Tank* (Version 1.7e) [Code] Available from: http://falstad.com/ripple/ [Accessed: 22nd September 2013]
- G-Truc Creation (2013) *GLM* (Version 0.9.4.5) [Code] Available from http://glm.g-truc.net/0.9.4/ [Accessed: 22nd September 2013]
- Lantinga, S. (2013) *SDL_ttf* (Version 2.0.11) [Code] Available from http://www.libsdl.org/projects/SDL_ttf/ [Accessed: 22nd September 2013]

- Lantinga, S. et al. (2013) *SDL* (Version 1.2.15) [Code] Available from http://www.libsdl.org/download-1.2.php [Accessed: 22nd September 2013]
- Sub Protocol (2013) *verlet-js* [Code] Available from: https://github.com/subprotocol/verlet-js [Accessed: 22nd September 2013]

Source evaluations

Graphics Gems Line Intersection algorithm

Date: 30/6/13

Resource: Goldman, R. (1990) Intesection of Two Lines in Three-Space. In Glassner, A. (ed.) *Graphics Gems* Academic Press Inc.

Description:

Objective:

I want to find out when a ragdoll limb collides with an obstacle.
In order to do this, I need to find out about how I could calculate whether two line segments intersect in 2D space.

Key points:

- I discovered a calculation that I could do in order to find whether two line segments intersect in 3D space.

Cross references:

The algorithm outlined in this source was referenced from one of my other sources, Stack Overflow, a community question-and-answer website.

Reliability:

The Graphics Gems book has a very good reputation in the graphics community for being of high quality.
In addition, I implemented the algorithm and it worked, which attests to this source's validity.

Future work:

I need to rewrite the equations in terms of 2D rather than 3D.

I need to implement the algorithm to test whether it works correctly.

Codeflow Verlet

Date: 17/6/13

Resource:

- Boesch, F. (2010) Hard Constraints, Easy Solutions. *Codeflow*. Weblog. Available from: http://codeflow.org/entries/2010/sep/01/hard-constraints-easy-solutions/ [Accessed: 23th September 2013]

Description:

This is a blog post written by Florian Boesch, which includes live demonstrations of a physics simulation method called Verlet integration.

Objective:

I wanted to find out whether fast realistic physics was possible to achieve with a computer simulation.

Key points:

The article was very useful in showing me that Verlet integration could potentially be what I am looking for:

- Physics simulation is feasible, even with a low amount of computation power available.
- Physics objects have to be modelled as particles joined together with "constraints".
- The Verlet integration scheme has been shown to be successful, being stable and realistic.

Reliability:

This is a blog post by a software enthusiast, so it in itself is not necessarily reliable. However, the live demonstrations that were on the page worked, and were convincing. I would use this blog post as a starting point for my investigations into Verlet integration.

Future work:

I need to investigate how exactly this was done, and the mathematics behind it. Although some code examples were given on the page, they were incomplete and not detailed enough for me to implement this myself.
I need to find other sources to verify that Verlet integration is suitable, and which kinds of constraints can be used with the integration scheme.

Thomas Jakobsen's Advanced Character Physics

Date: 25/06/13

Resource: Jakobsen, T. (2001) Advanced Character Physics. *Game Developer's Conference 2001* Available from: http://web.archive.org/web/20080410171619/http://www.teknikus.dk/tj/gdc2001.htm [Accessed: 22/09/13]

Description:

This is a paper written by Thomas Jakobsen, published for the Game Developer's Conference 2001. Jakobsen is a professional mathematician and programmer, who wrote the physics engine for the videogame "Hitman: Codename 47", which revolutionised how ragdoll physics would be calculated in future games. Jakobsen popularised the use of Verlet integration in order to drastically reduce the amount of computation power required to simulate a ragdoll, and increase the realism and stability of the simulation at the same time.

Objective:

I was hoping to find out about techniques I could use to accurately simulate physics.

Key points:

- Verlet integration is not 100% accurate, some energy can leave the system.
- The core mathematics required is introduced.

Reliability:

This is a primary source, written by a leading programmer himself about his own creation. I have found many online sources that refer to and cite this publication. This leads me to trust this source as relevant and reliable.
In addition I have tried the technique outlined in the paper in my "Verlet" experiments, and they have all been very successful.

Cross-references:

I arrived at this source after it was referenced as the fundamental Verlet integration source from several places across the internet. The information in this source connects with the information I had found in the Codeflow blog post and the Democritus introduction to Verlet, as they agree on the key underlying formula.

OpenGL Programming Guide (Red Book)

Date: 13/08/13

Resource: Shreiner, D. et al. (1997) *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.1*. 2nd edition. Addison-Wesley Publishing.

Description:

This book is the official guide to OpenGL. It contains tutorials and examples on the key parts of OpenGL and graphics programming.

Objective:

I was aiming to find out about "matrix stacks": what they were and how I could use them to represent transformations on groups of objects in 3D space.

Key points:

- I discovered what matrix stacks were.

Reliability:

This is an official guide written by the OpenGL Architecture Revision Board. Because of its official status, the information should be reliable.

Cross-references:

The code examples in the book did not match up with those in other sources, such as McKesson's OpenGL tutorial. I discovered that this is because the code in this source was written using a deprecated style called immediate mode. Modern OpenGL no longer uses this, so I ignored the code examples in this source and focused only on the concepts and mathematics.

"arcsynthesis.org" OpenGL tutorial

Date: 1/1/13, 23/1/13, 25/1/13, 18/9/13

Resource:

- McKesson, J. L. (2012) Learning Modern 3D Graphics Programming. Available from: http://www.arcsynthesis.org/gltut/ [Accessed 23rd September 2013]

Description:

This online tutorial teaches the basics of graphics programming. It uses OpenGL in its code examples, however it is not designed to be a tutorial on OpenGL specifically. It covers a wide variety of topics, including perspective transform, camera and world matrices, and lighting.

Objective:

To find out about how graphics hardware works, and how OpenGL interfaces with it.

Key points:

- I learnt about the structure and architecture of graphics hardware in modern computers.
- I learnt about concepts in OpenGL that would lead to efficient 3D rendering.
- I learnt about lighting and fragment shading.

Reliability:

- This source has been cited by many places (e.g. Stack Overflow and the gamedev forums) as one of the best up-to-date tutorials on graphics programming.
- I am slightly worried that I am overly dependent on this source, as it contains so much useful and relevant information. In order to combat this, I am checking and comparing all the information that I find in this source with other sources.

Cross-references:

It broadly agrees with other sources, including the OpenGL code API and other gamedev.net tutorials on lighting. However in some places it takes a slightly different approach. As this source seems to be the most reliable and had the most explanations behind its approach, I used this as my primary source of information when there was a conflict.