2005-01-01

Learning Games Programming With Dalek World.

Bryan Duggan
Technological University Dublin, bryan.duggan@tudublin.ie

Hugh McAtamney
Technological University Dublin, hugh.mcatamney@tudublin.ie

Fredrick Mtenzi
Technological University Dublin, Fredrick.Mtenzi@tudublin.ie

Recommended Citation

Duggan, B., McAtamney, H. & Mtenzi, F. (2005). Learning games programming with Dalek World. *7th. International Conference on Computer Games: AI, Animation, Mobile, Educational and Serious Games. CNBDI*, Magelis, Angouleme, France, 28-30th. November. doi:10.21427/k45z-fs81

Learning Games Programming with "Dalek World"

Bryan Duggan, Hugh McAtamney,
Fredrick Japhet Mtenzi
School of Computing,
Dublin Institute of Technology,
Kevin St., Dublin 8.

Email: {bryan.duggan, hugh.mcattmney, fred.mtenzi}@comp.dit.ie
Web: http://www.comp.dit.ie/bduggan
Phone: + 353 1 4024788

Keywords

Computer Games, Education, Course Curriculum, Object Orientation, Design Patterns, Tools, Frameworks.

Abstract

From September 2005 the School of Computing in the DIT will offer an elective in computer games programming to final year computer science students. This paper demonstrates how students will learn games programming by developing a 3D FPS (First Person Shooter) called Dalek World. Dalek World is developed using Microsoft Visual Studio in C++ and was originally developed by the authors to learn games programming techniques themselves. In developing Dalek World, students will learn Euclidian geometry, 3D graphics programming, object orientated game design, level loading, physics, collision detection, the A* algorithm, perception and enemy AI using finite state machines.

1 Introduction

From September 2005 the School of Computing in the DIT will offer an elective in computer games programming to final year computer science students. The course will run for four hours per week over two semesters. In semester one, the emphasis is on understanding how a game engine is programmed. This includes an understanding of Euclidian geometry, object orientated game engine design, level loading, collision detection, physics, the A* algorithm, perception and enemy AI using finite state machines. Semester two will cover scripting, modding, 3D modelling, level design, game balance, content creation, roles in games development teams and gaming and society. Semester two uses the industry leading sandbox engine (from the bestselling game FarCry) as a tool set. Students taking the elective will have previously studied the C++ language for three years. Students may also choose complementary electives such as computer graphics and artificial intelligence and so these topics are not covered in depth on the course.

In preparation for teaching the course and based on the premise that the best way to master a new discipline is to become a practitioner, we developed an FPS entitled Dalek World in C++ over the summer of 2005 [Kolb, 1983]. The game was developed using standard tools including Microsoft Visual Studio and the DirectX SDK. Our goals in developing the game were twofold. Firstly, although we have been keen computer game players for many years and have much experience developing complex software, prior to this summer we were generally uninformed about the tools and technology used to create 3D computer games. Our second reason for developing the game was so that we could develop useful and relevant learning material for students taking the games programming elective.

The game is a typical (though simple) first person shooter that pitches the player against computer controlled enemy daleks. A *dalek* is a fictional extraterrestrial mutant from the British science fiction television series Doctor Who [Wikipedia, 2005].

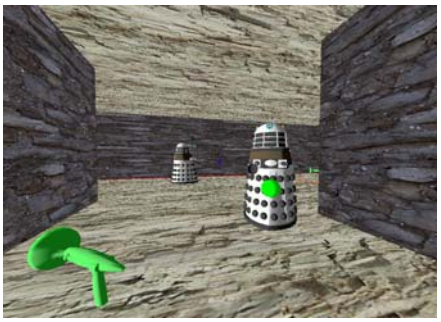

Figure 1: A typical screen shot from "Dalek World"

Dalek's were chosen to be the enemies in the game because they are recognisable by students due to their appearance for many years on television and also because a dalek has no

limbs and hence does not require animation to move. Figure 1 illustrates a typical screen shot from Dalek World. In this screenshot are shown a number daleks firing projectiles at a player in the virtual world.

It is our plan to provide a guided "studio classroom" experience to students taking the elective so that they can develop this game independently over the thirteen weeks of semester one [Carbone & Sheard, 2002] [Mahar & Lahart 2004]. Some of the code (such as a basic 3D graphics framework) will be provided, while other parts of the code students will program themselves. [Livingston & McMonnies, 2004] suggest that a C/C++ framework might be developed as a tool to teach games programming. Our experience developing a course around Dalek World suggests that it is a suitable framework.

This paper describes how the development of Dalek World by students is broken down into a series of goals with specific learning outcomes. Section 2 describes how students learn Euclidian geometry, 3D graphics programming and object orientated game development by creating a world for game entities to inhabit. Section 3 describes how students learn the basics of collision detection by implementing bounding box and bounding sphere collision detection in the game. Section 4 describes how students learn to implement Newton's laws of force and acceleration to move daleks around in the game. Section 5 describes how students learn path following and the A* algorithm by solving the problem of how daleks can find ammunition. Section 6 describes the perception algorithm students develop. Section 7 describes how students program intelligence into the daleks using finite state machines and section 8 discusses future work. This paper is of interest to academics developing games programming courses and to others wishing to get a concise overview of the technologies and algorithms used in 3D games.

2 Building the Game World

The first task students approach is to build a 3D environment in which the player and the daleks can move around. The first weeks of the course are used to introduce students to Euclidian geometry, vectors, trigonometry and matrices. Using these mathematical tools, game objects can be positioned, orientated and moved in 3D. The goal is for students to create a "level loader" for Dalek World. Additionally students derive the mathematics to program a camera in 3D that can walk, strafe, yaw and pitch using keyboard and mouse control. This involves deriving the vector and matrix mathematics required to program the camera.

Students are also introduced to an object oriented approach to game engine development as illustrated in the Dalek World framework classes in Figure 2. These classes are gradually introduced and enhanced as the course progresses and additional functionality is required so that by the end of this section, students have a complete object orientated framework for the game. The level loader that students develop loads required assets (such as meshes, textures and materials) and sets up initial object geometry. Assets are managed by an `AssetManager` singleton class. In this way, students are introduced to the singleton design pattern and lazy initialisation.

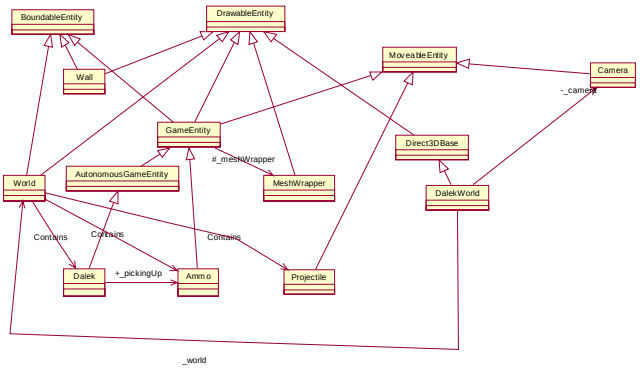

Figure 2: Class diagram of the Dalek World game entity classes

Environment functionality is encapsulated in the `World` class (which extends `Drawable`). The world class loads the geometry for the world, walls, initial positions and patrol points of daleks, weapon spawn points, mesh and texture file names from a text file. This provides the flexibility to change the world by editing a text file. This is analogous to "modding" Dalek World. By using different world files, students can add and remove daleks and weapons and change the game layout. In developing the `World` class students are introduced to the concepts of vertex buffers, texturing, world transformations, and the STL. Students must also derive the mathematics to position and orientate game entities using vector operations (+, -, *, vector dot and cross products) trigonometry and transformation matrices. Position and orientations are stored as `D3DXVECTOR3` data structures. These structures hold x, y and z co-ordinates for the entity and a unit vector of the orientation. For simplicity, an entity in Dalek World always "moves" and "faces" in the direction pointed to by its orientation vector also all walls are orthogonal. This simplifies the collision detection calculations students must derive later.

3 Collision Detection

This is followed by adding collision detection to the game. Collision detection in computer games is a non-trivial problem and would require many weeks of class time to be covered in detail. For simplicity therefore, in Dalek World, collision detection is limited to detecting if a player movement will cause a collision with a wall or a dalek and to check to see if the path of a projectile collides with the player, a dalek or the world. Although the algorithms developed by students are simple, an object orientated approach is again taken, in that a class that wants to implement collision detection can inherit from the `BoundableEntity` class. Students learn functionality to add collision detection by using DirectX API's for calculating bounding boxes and bounding spheres for meshes and implementing *inner bounding box*, *outer bounding box* and *bounding sphere* collision detection in the `BoundableEntity` class.

4 Physics

Newton's Laws of force and acceleration can be used to move characters in computer games in a realistic manner. In this section of the course, students learn how to program these laws in Dalek World to move the daleks and to fire projectiles.

Daleks extend the `AutonomousGameEntity` class (which extends `GameEntity`). `AutonomousGameEntity` holds the entity's mass, maximum speed and maximum force applicable in addition to having a pointer to a `Locomotion` object. The Locomotion class is adapted from [Reynolds, 99] "steering behaviours" and calculates the force required to move entities in the game. `Locomotion` calculates the force required to move the `AutonomousGameEntity` and multiplies it by the time delta since the last frame update. This force is applied to the entities velocity vector, which is used to update its position. The velocity is then normalised to create a new heading vector for the entity, so it can be orientated to face the direction in which it moves. To develop Dalek World, students need to implement just three steering behaviours: *seek*, *arrive* and *follow_path*. An introduction of physics programming is particularly relevant, given that the physics engine for many popular games is developed by Irish company Havok [Havok, 2005].

5 Navigation

Daleks in the game need to navigate along their "patrol" paths and to navigate from arbitrary positions in the environment to weapon spawn points.

Students first learn how to implement path following by creating a `Path` class and using vector mathematics to make daleks follow the path. Students then learn to calculate lowest cost paths from daleks to ammunition spawn points using the A* graph traversal algorithm. The A* algorithm is used by almost all commercial games to calculate paths. The material used to teach the A* algorithm is adapted from [Lester, 2005].

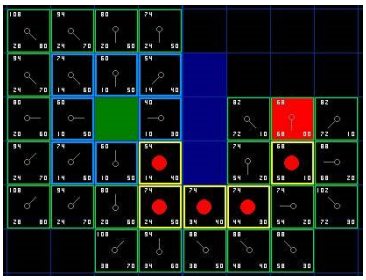

Figure 3: Learning the A* algorithm

Figure 3 is an illustration adapted from [Lester, 2005] that illustrates a lowest cost path from the node on the left to the node on the right avoiding the wall in the centre. Students learn to use the two most common A* heuristics, the Euclidian distance and the Manhattan distance. When a dalek follows a path, the path through the graph can be drawn in red as illustrated in Figure 4.

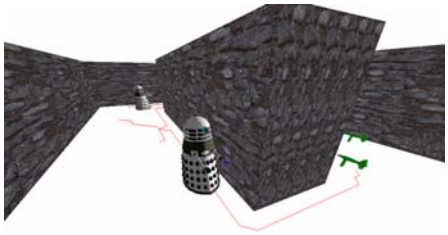

Figure 4: Dalek World with path drawing enabled

6 Perception

Daleks need to detect if they can "see" the player. Students learn to implement *perception* as a two stage process as illustrated in Figure 5.

Stage one is to calculate if the player's position vector is within the field of view of the dalek. The second stage of this calculation is to calculate if there is an unobstructed line (of sight) between the dalek and the player. To achieve this, students develop code to compute the intersection point between the line from the dalek to the player and lines representing each of the sides of each of the walls in the world.

This algorithm works in a 2D world, however as all the action takes place in Dalek World on one "plane" (0 on the y-axis), this algorithm works well and enables students to develop the algorithm in a two hour lab class.

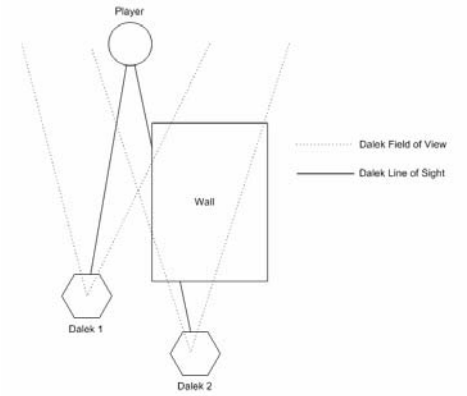

Figure 5: Dalek perception

7 Decision Making

In this section of the course, students learn to control each dalek using a set of "states". State machines are used as to implement AI in the majority of commercial games, such as FarCry, Half Life 2 etc.

The state design pattern [Buckland, 2004] is used here to encapsulate state transition logic within the states themselves. For example the `SearchAmmo` state uses the A* algorithm previously learned by students to calculate a path from the daleks current position to the nearest ammunition spawn point. This is calculated in the transition to the state. Figure 6 is a state transition diagram for the daleks in Dalek World.

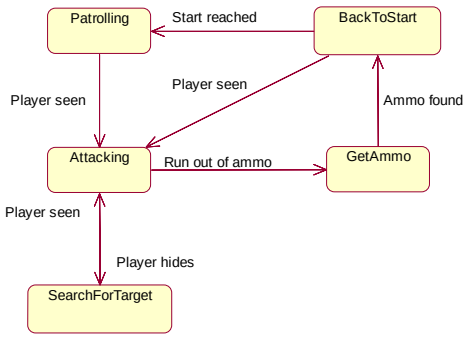

Figure 6: State Transition Diagram of Dalek World

To conclude semester one by learning to give the daleks "autonomy" and "intelligence" means that at this stage, students possess the core skills required by developers in the games industry in addition to increasing their knowledge of vector and matrix mathematics, advanced C++, templates, object oriented development, physics, graph theory and AI.

8 Conclusions and Future Work

This paper described a games programming course based on Dalek World. In conclusion, the development of Dalek World by the authors proved to be a useful framework for learning the concepts of 3D game programming. In programming Dalek World, it is hoped that students similarly will learn Euclidian geometry for games programming, an introduction to 3D graphics, object orientation, physics, path finding, perception and the state machine design pattern.

Many game features remain to be implemented. These include:

- Dalek and player health. Projectiles currently do no "damage" and players and daleks cannot "die".
- Daleks can pick up additional ammunition, but because daleks cannot be damaged, there are no health spawn points.
- All walls are orthogonal. Non-orthogonal walls could be added.
- Perception/collision detection could be enhanced by adding ray/plane intersections. This would facilitate the creation of more flexible world features such as multi-storey levels, stairs etc.
- AI could be improved with the addition of additional states and transitions. For example, if a dalek is hit with a projectile, it should turn and attack its attacker.
- 3D positional audio could be added.
- Scripting could be added by integrating the Lua open source scripting language. This is currently planned for semester two.

These additions will form the basis of student tutorials assignments and projects over the course of the semester. In future work we hope to present student experiences on the Dalek World based course. It is further expected that lessons learned during the course might lead to excellent final year projects in the area of games programming.

The course has proven extremely popular, being chosen by over fifty percent of students this year (2005-2006). This popularity has demonstrated that not only has the course the potential for learning skills required to develop games, but also that it can be used as a gateway for developing advanced skills in computer science in general. It is hoped that elements of games programming can be integrated into other subjects of the degree. Possibilities include the teaching of networking using online multiplayer games, ubiquitous technology using mobile gaming and programming using the DarkBasic games programming language [Livingston & McMonnies, 2004].

References

[Buckland, 2004] M. Buckland: *Programming Game AI by Example*. Wordware Publishing, Inc., November, 2004

[Carbone & Sheard, 2002] A. Carbone, & J. Sheard: Developing a Model of Student Learning in a Studio-Based Teaching Environment, Informing Science & IT Education Conference, Cork, Ireland, June 2002

[Havok, 2005] http://havok.com, Accessed September 2005

[Kolb, 1984] D. A. Kolb: *Experiential learning: experience as the source of learning and development*, Englewood Cliffs, N.J. : Prentice-Hall, 1984

[Lester, 2005] P. Lester: A* Pathfinding for Beginners, http://www.policyalmanac.org/games/aStarTutorial.htm, Accessed October, 2005

[Livingston & McMonnies, 2004] D. Livingston & A McMonies: Is DarkBasic to be considered harmful?, 5th Game-On International Conference, Reading, UK, November 2004

[Luna, 2003] F. D. Luna: *Introduction to 3D Game Programming with DirectX 9.0*. Wordware Publishing, Inc.; 1st edition, June 2, 2003

[Mahar & Lahart 2004] E. Mahar, & O. Lahart: Student Centered Learning in a Studio Classroom Environment, EdTech, Tralee, Ireland, June 2004

[Reynolds, 1999] C. W. Reynolds: Steering Behaviors For Autonomous Characters, in the proceedings of Game Developers Conference 1999 held in San Jose, California. Miller Freeman Game Group, San Francisco, California.

[Wikipedia, 2005] Dalek, http://en.wikipedia.org/wiki/Dalek, Accessed September 2005