

A Black Box Exploration: Football Game Graphics

Gregory S. Nilsen
Computer Science Department
Hiram College
Hiram, OH 44234
nilsengs@hiram.edu

Abstract

In the world of video and computer games, each year we expect a new level of realism to be added to the visuals of the game. However, we often neglect the difficulties of doing so because the details are neither obvious nor available to us. Therefore, the only way to learn about them is to attempt to "recreate" them ourselves.

In this project we will explore the methods of creating an American football player in OpenGL including topics such as construction, lighting, shadows, texture mapping, and animation. We demonstrate the potentials of such a representation, as well as the difficulties in creating it.

Keywords

OpenGL, graphics, football, animation, lighting, projection shadows, texture mapping.

Site Location

<http://cs.hiram.edu/~nilsengs/football/>

1. Introduction

One of the mediums where graphics is most visible in the past two decades has been in the video game. We have seen Pong, Donkey Kong, and numerous other titles come and go. Each step along the way, a new level of realism has been introduced in order to enhance the game playing experience.

Today, this realism has reached astonishing levels, but as consumers we often neglect how far the game graphics have really come. However, the video game companies know this all too well, and keep their innovations under tight wraps for commercial reasons. Therefore we are put in a position for black box testing, where we must attempt to discover these techniques on our own.

In order to learn more about video game graphics we attempt to "recreate" the game. In this project, we specifically explore the methods of creating an American football player in OpenGL with the use of the OpenGL Utility Toolkit (GLUT). This is done using human figure modeling, realistic animation of the created figure, and then adding detailed effects such as lighting, shadows, and textures.

This project was taken on for several reasons. First, we are football fans and are familiar both with the real sport and the video games that it has inspired. This familiarity has encouraged us to continue to learn more about how the games are created.

Secondly, it is an extension of a previous work in artificial intelligence [10]. In this project, we created an intelligent defense that reacted to a pre-programmed offense. Unfortunately, that program was created solely using text output to represent the field; a representation that was rigid and unrealistic. A more authentic graphical representation was desired and therefore worked on with this project.

Lastly, this project was also inspired by a project done by Mike Johns at Hiram College in 1998. He created a project that placed a graphical interface on a previous hockey project in artificial intelligence, called mHockey [5]. This project has inspired some of the things we have done in this project, and none more than attempting to combine graphics with an existing project in artificially intelligent game play.

2. Background Information

In order to complete this project, several areas of advanced graphics techniques were investigated. The first of these topics is texturing. Texturing can be done in a number of different ways, and they are all computationally intensive when using graphics not embedded within the code of the project. One method that we found was using the raw portable pixelmap (raw PPM) image format as specified by Nate Robins [12]. We could then import the image and specify where in our environment it should be displayed. Details of this process will be discussed later in this paper.

A second topic is shearing, a transformation that causes an object to be slanted to the side. An example of this would be to take a Slinky toy. When the toy is sitting on one end, it is a regular cylinder. We can also then push the top to the side so that it is not directly over the base of the toy. This is the effect that shearing gives us.

This shearing effect is accomplished by creating a new matrix specifically for shearing. This matrix contains values that specify in which direction an object should be sheared, and by how much relative to the size of the object in another direction. The current matrix must then be multiplied by this shearing matrix in order to create the shearing effect [9].

The third of these topics is shadow creation. In our research, we found that there are many methods used to create

shadows. The most realistic of these is to pick a point being displayed in the window, and create an unseen line between that point and the light source. Then, the line is checked to see if there are any other objects intersecting the line. If there are, then the original point is drawn darker to represent a shadow. However, this approach is very computationally intensive [7].

Another method we discovered used what are called projection shadows. Using this method, the object is redrawn in black, and then sheared away from the light. The sheared object is then scaled so that it has no height. This gives the perception of a shadow. However, this only works with flat surfaces because the shadow object does not conform to the object it is being “projected” onto like an actual shadow would [2].

Since the human figure is a very complex entity, the modeling of its movements can prove to be a difficult task. For this reason, research had to be conducted about the physics of movement in order to obtain realistic movement for our figure. One particularly useful item was a video clip on passive walking [8]. This demonstrated the motions that are made by the legs in order to walk. This information could then be used to create a model for running.

One last area that had to be looked into was the rules and regulations of American football [11]. Even the most rabid football fan usually does not know all the dimensions of the field, so this information had to be found to create a scale model. We were then able to take all this information and put it to use in our creation of the football player model.

3. Related Work

In our investigations, we found two projects that were doing related work. The first of these is the mHockey project by

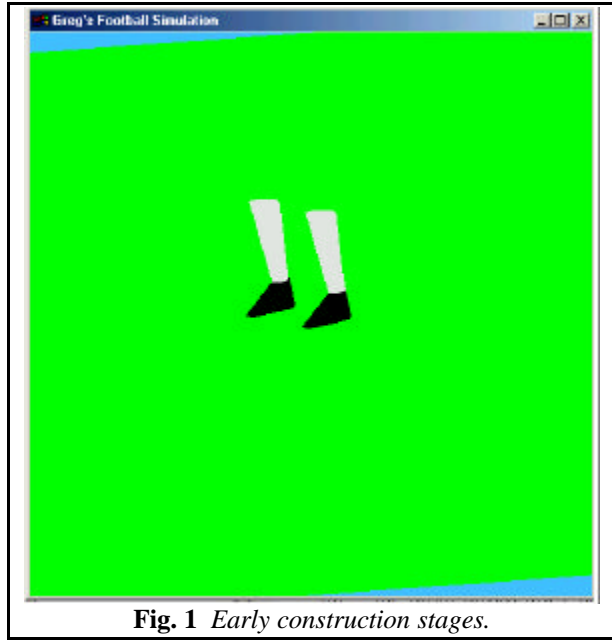


Fig. 1 *Early construction stages.*

Mike Johns that was previously mentioned [5].

This project has undergone several revisions since the original creation, and continues to be improved upon to this day. His work in player modeling and texturing are very similar in design to the player that we have created. This project helped to give us an idea of what could be done with our own project and it is a very similar project, but with the sport of hockey as its main focus as opposed to football.

Another similar work comes from the Slovak Institute of Technology. Here, Roman Filkorn and Marek Kocan have written a work entitled “Simulation of Human Body Kinematics” [3]. This work explores the simulation of human movement using OpenGL, VRML and Java. This resource was particularly useful for our modeling of the football player because we could better analyze how realistic the movements actually were.

4. Project Description

When this project was first approached, we attempted to break it up into several stages. However, these stages were changed and adapted as the project progressed. In

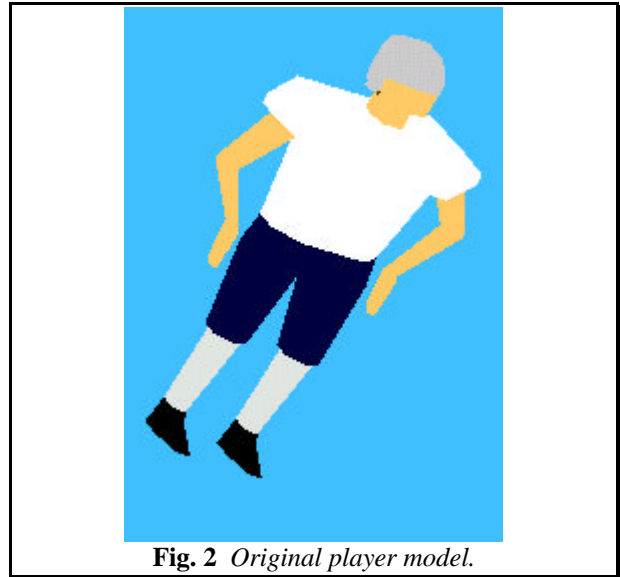


Fig. 2 *Original player model.*

the end, we saw that there were five distinct phases of this project: construction, lighting & materials, animation, shadows, and texturing.

4.1. From The Ground Up

In the first stage of the project, we began to build the figure of our football player. We literally did this process from the ground up, as can be seen in an early screen capture of the figure [Fig. 1]. At this point in the construction, there were only calves and a pair of feet built on a green plane to represent the field. These objects were created using the polygon constructor in OpenGL, and the rest of the player was built in this fashion. The colors were determined using the `glColor3f` function to explicitly designate colors for the sets of polygons that made up the figure.

One decision that we made when constructing our player was to set up our world coordinates in an unusual way. Instead of using the typical setup (where the y-axis is the vertical axis, the z-axis runs in one direction on the origin “plane”, and the x-axis runs in the other direction), we felt it more useful to use the z-axis as the vertical axis. This is due to the fact that our artificial intelligence project represented the field in two dimensions. We could then use the x- and y-axes to represent the position of the

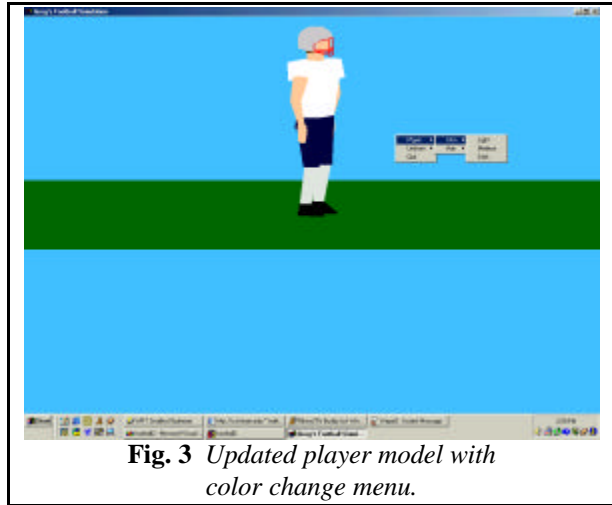


Fig. 3 Updated player model with color change menu.

player on the field in a manner similar to that project. Therefore, we set the x-axis to run in the direction of one sideline to the other, while the y-axis runs in the direction of one end zone to the other. This would allow us to more easily combine the two projects at a later point in this graphics project.

Throughout the construction process, the entire figure was built using the polygon constructors [Fig. 2]. In this screen capture, the field has been removed and the keyboard has been configured to rotate the player around the axis. This was done so that all angles of the player could be seen and checked for correctness. Also, the facemask has not yet been included at this point but was added before long.

When we were constructing our figure, we began with a semi-hierarchical structure. What we mean by this is that the parts of the body had a positional dependency upon one another, but they were not properly reliant when it came time to add motion. When the features of the hierarchical system were discovered, the model was then properly adjusted to take advantage of them.

Also added at this point, a menu bound to the right mouse button was added. This menu contained a listing of player color attributes that could be changed by selecting the desired color. For example, the user could choose to change the players skin

shade as can be seen in a late construction screen capture of the figure [Fig. 3]. Using variables as parameters in the glColor3f function and changing their values through the menu functions allowed us to change the colors. This was added to allow the user to replicate their favorite team if they so desired.

The next items that were added to the project were the field elements. We started by adding a ball that could rest in the right hand of the player. We then added goalposts, with all the proper dimensions as specified in the National Football League rules [11], to each end of the field. This was done to prepare the field for actual play when the project was finally combined with our existing artificial intelligence project. The objects also add to the realistic effect because the sport of football is played on a specifically designed field, not just an open field as an empty plane may suggest.

Once we completed the original model of the player, we learned of a feature of OpenGL and GLUT that became particularly useful: the primitive shapes [6][13]. With our earlier model, we had begun calculating the normal vectors we were going to need for the lighting portion of the project. However, we learned that with the primitive shapes, we could have these normal vectors calculated for us.

Therefore, we went back and rebuilt the forearms, upper arms, thighs and calves of the player using the cylinder primitive. In order to get the correct angles for the arms, we sheared them in the x-direction. The biceps were sheared away from the body, while the forearms were sheared back towards the body. This gave the figure an appearance similar to a person resting their hands on their hips.

However, we could not use primitive objects for the entire player. Because of the odd shapes of the feet, waist, torso, head and helmet, these items continued to be created

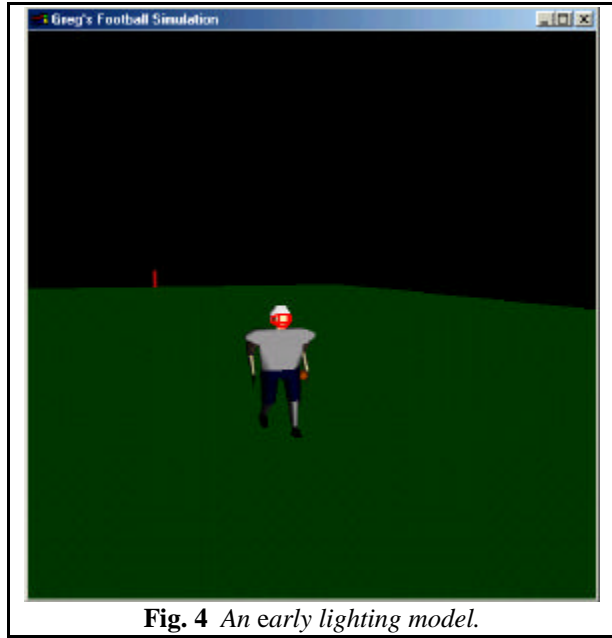


Fig. 4 *An early lighting model.*

by polygon constructors. The helmet was a particularly difficult decision, because we wanted to give a more rounded appearance to it, but we could not use a primitive object such as a sphere for it because we could not “cut out” the front part to see the face. So even though the appearance was not exact, it gave a good representation, especially when we got to the point of adding lighting.

4.2. Lights! Makeup!

Once we completed the construction of our player model, we still did not have a very realistic representation of an actual football player. One of the main reasons for this is that the player did not appear to be in three dimensions because there was no light and shading to create the perception of depth. Due to this problem, our efforts were turned to the next stage of the project: creating realistic lighting.

In our first attempt at lighting, we took some demonstration code from the Interactive Computer Graphics textbook written by Edward Angel [1]. This code contained information for the lighting and materials. However, when we ran the program, everything turned red. The problem we found was that we had not defined our materials properly.

Once that discovery had been made, we went ahead and replaced all the `glColor3f` functions with materials definitions. These definitions maintained their colors using the variables that had been used previously, but also added the other necessary properties of the material. These properties include how much of each type of light the material will reflect and how shiny it should appear. At this point, we were using a single point light source in an attempt to create a sunlight effect. This combination of effects gave us a more realistic image but we were still having troubles with some of the materials.

During the construction process, we defined all the normal points for the polygons. However, when the lighting was applied, those polygons appeared more like neon lights. This effect can be seen in a screen capture from one of our early lighting attempts [Fig. 4]. We then discovered that our normal vectors were being properly calculated the first time, but were not being updated. This caused the materials to reflect light in an unnatural manner. This problem was corrected by enabling OpenGL to normalize the vectors so that the correct values were being used.

Another difficulty we had was the plastic-like appearance of the materials. Since we had three different types of materials (plastic for helmet, cloth for uniform, and a skin material) we did not desire them to all appear like plastic. In order to get a more realistic representation, we experimented with different material properties. In the end, we found that most materials do have a plastic look when using OpenGL. However, we could make some materials look less like plastic than others.

An additional change that we made from the original lighting model was adding a second light source. A single light source caused a lot of shading on the field and the player, and this did not give us the sunlight effect that we desired. We then

experimented with a second light source that was placed on a different side of the original position of the player. We discovered that when it was combined with the first source, they worked together to give us an effect that was very similar to that of sunlight. Therefore, in the final lighting model, two light sources were used. Also added was a control in the existing right mouse button menu that allows the lights to be toggled on and off.

4.3. Off And Running

After we had created a more realistic looking model for the football player, it was time to get the project moving. Literally. We took the player model that we had and then added functions that simulated the basic motions needed to play football: running, jumping, diving, throwing, catching and blocking.

As noted above, we realized before we began creating our animations that a hierarchical model was going to be needed for the motions. This is due to the fact that body parts rely on the parts connected to them for their movement. For example, the forearm cannot move around a point on the shoulder unless the upper arm moves the forearm.

Though each of the motions we created has a quirk of its own, they can all be done using rotations and translations. This is a very useful property because OpenGL contains support for both these transformations. We then placed these transformations into while loops, which allowed us to simulate the movement of the player.

We also needed to create a new group of variables that would hold the rotation directions and amounts. These allowed us to easily increase and decrease the amounts of rotation and translation. Though the number of variables was great, they helped organize the movements so that each one could be

thought of individually, and then later combined into one fluid motion.

The first of these motions that we attempted was running. We began this by researching the mechanics of running and walking. We found a particularly useful web site containing research on “passive walking”, including video clips of mechanical walking models. We then were able to model the running of our figure by adapting from these models and experimenting some to find animations that looked real.

While attempting to create the realistic running effect, we found that the arms of our player appeared in an awkward fashion. Therefore, we began experimenting to find a more natural looking position for the arms. We finally decided to rotate the arms from their original position with the elbows pointing outward; to having the elbows point straight back. This gave a more relaxed and realistic look to the figure as a whole.

When we came to the throwing model, we ran into more difficulties. One reason for this is that very few places research the throwing motion of a football quarterback and its potential application in graphics. Therefore, we took an observation approach, and watched our own motions when throwing a ball. This model allowed us to still create a very realistic motion even though other research was not available.

Another problem with the throwing motion that we encountered was in order for the player to throw the ball; it must be represented as an independent structure. Up to this point the ball had been bound to the right hand. We then decided to have the ball bound to the hand when the player was in possession of the ball, but to allow the ball to move independently around the field when it was thrown, or on in the air before being caught.

Unfortunately, while we were working on the running, jumping and diving models,

we did run into some translation problems. We would increment the position of the player by small amounts in the while loop for the motion, but the player would always “stagger” forward when running, and upward when jumping and diving. This caused the animations to look much less realistic. However, after extensive experimentation and testing, a solution to this problem has not yet been found. Instead of lingering on the problem, we continued to work on other aspects of the project while continuing to brainstorm new solutions to the problem.

4.4. The Shades Make The Look

Once we had completed our animation process, we went on to fix an almost glaring problem with our project at the time. That flaw was having lighting, but no shadows. Every day we see shadows of objects and hardly think anything of them, but when we see an object that is has lighting on it, but no shadows it does not look realistic. We had this problem with our project, so we decided to find a way to correct that.

After researching different methods of creating shadows, we decided to create our shadows using projection shadows as specified by David Blythe and Tom McReynolds [2]. As mentioned above, this method does not work on surfaces that are not flat. However, we were working with a single plane as our field so this method was viable.

Once we had the code that drew and transformed the new player shadows (one for each light source), we took special care to make the shearing of the shadows realistic. We first drew tiny spheres where the light sources were located on the field in order to better visualize the movement of the shadows. We then calculated the direction of the shear and the length of the shadow based on the location of the player on the field and the position of the light in the created world. This allowed us to create

shadows that, if the player were to run around the light source, would move appropriately. In the end, the shadows behaved fairly realistically.

However, at this point the two shadows were a very dark black. This is a very unrealistic effect with two light sources because there is no lightening of the shadow due to the interference of the other light source. In order to create this effect, we adjusted the alpha channel to make the shadows lighter. However, this did not change the appearance of the shadow to our surprise.

Instead, we had to also enable the `glBlendFunc` that would blend the material that is being made translucent by the alpha channel together with the field that the shadow is being projected onto [4]. When this property was enabled we were able to get much more realistic looking shadows for the scene.

As we mentioned in the section on lighting, we enabled a toggle to turn each of the light sources on and off. Therefore, when we created shadows to go along with the lights, the shadows should be toggled concurrently. Therefore, we set up the program to only display shadows for light sources that are on. We also increased the value in the alpha channel of the materials when one light source went off to create a darker shadow since there would be no interference from another light source.

By this point in the project, we had created a rather realistic looking player model with animations and proper lighting effects, but there was still something that looked wrong with the field. Therefore, we went on to make the field look more realistic.

4.5. Paint Job

At this point in the project we began to address more of the cosmetic details. One of the problems we had was that there was a field with goalposts, but no end zones or

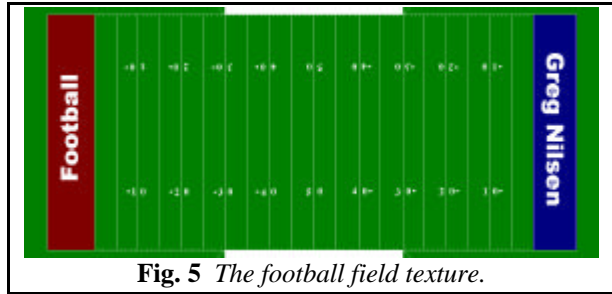


Fig. 5 *The football field texture.*



Fig. 6 *The player face texture.*

yardage lines. We also had a player without a face that we wanted to correct.

One option for fixing these problems would have been to create more polygons or primitive objects to represent the field markings and the face of the player. However, this would have been a very extensive process with a large number of objects and transformations to place them.

Instead, a more commonsensical method involves placing an image on the desired surface. We started this process by creating a pair of bitmaps to use, a field complete with markings and a smiling face [Fig. 5, Fig. 6]. We then went out on the Internet in search of a method to place our newly created bitmaps into our project. We came across a set of tutorials created by Nate Robins that contained one specifically on texturing. From this tutorial, we borrowed a library that he had created for reading the raw PPM format into a project [12].

The problem we then had was that we had no direct way to convert our bitmaps to the raw PPM format on the Windows platform. We then began searching the Linux machines that we have in our department and came across the program xv. This program then allowed us to directly convert the bitmaps into a raw PPM files. We then took the new images and inserted them onto the field and the face of the player.

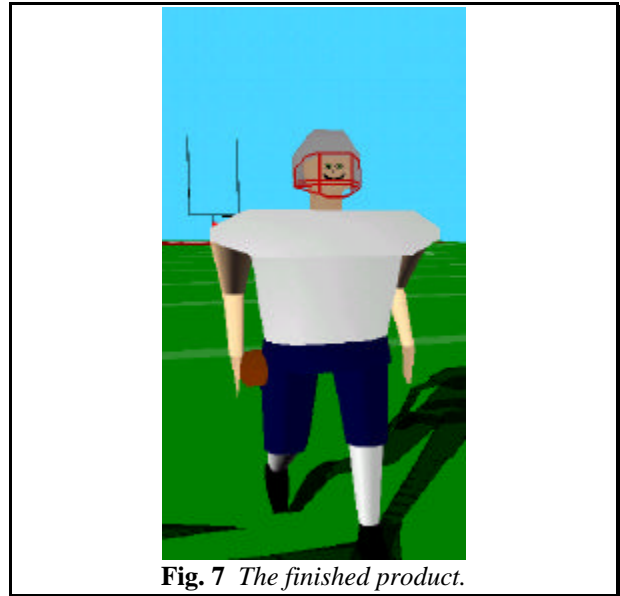


Fig. 7 *The finished product.*

As we mentioned above, we have included two textures in this project, as can be seen in the screen capture of the final product [Fig. 7]. Alas, this has caused a problem. When just one image is used, that image can be read into the program just once per execution and then it can be reused when the image is redisplayed. However, when multiple images are used, they must be read into the program every time they are displayed. This turns out to be a very computationally intensive process and it slows down the execution of the program greatly.

Therefore, we have also included toggles to enable and disable each texture in the right mouse button. This allows the user to prevent the program from loading the textures every time the screen is updated, and allows the program and the animations in it to execute much faster.

5. Evaluation of Results

Upon completion, we feel that our project has turned out very successfully. We were able to work with several advanced graphics topics including shadows and texturing. We were also able to work with other advanced topics such as the modeling of human movement. Even though we did not

meet our original expectations; we feel that this project was still successful.

This is because our original plans were very large in scope. We had hoped to create two teams of players who would play against one another using the artificially intelligent methods that were developed in our previous project. We had also hoped to do much more texture mapping and player interaction. However, all these tasks proved to be too much to accomplish in the twelve-week time period of the project.

So, compared with our original plan, we only completed about half of what we wanted to. One reason for this is that we worked with more advanced topics than we originally thought we would encounter. We also found that the animation of the figure absorbed much more time than we had anticipated. A final reason is that the original scope was enormous. However, much of that results from not knowing what is and is not capable of being done with the OpenGL environment we worked with.

Because of the difficulties we encountered with these topics, we had to change our plans over the term. Instead of being a game as originally anticipated, the project became an examination of creating realism in video gaming. Despite the change, it has still been an experience that we have gained much knowledge from.

6. Future Research

If we were to continue our research from the current state of the project, we would like to continue to work towards the completion of our original plan. We would like to include such aspects as the creation of multiple players who are capable of moving independently, the ability for players to interact, and to eventually complete the combination of this project and the one previously created for an artificially intelligent football defense.

7. Conclusion

As we have seen here, the creation of realism in video gaming is not a simple task. Even without having the code for the games we see on the market today, we can still see that texturing, animation, and lighting that they include are not simple processes. This project has successfully created a small portion of a football game and shown the difficulties in doing so. Overall, this project has offered a great amount of insight to a world of programming that is appreciated, though not always understood, by the people who enjoy video games all over the world.

References

- [1] Angel, Edward. Interactive Computer Graphics. 2nd Ed. Reading, MA: Addison-Wesley Longman, Inc., 2000.
- [2] Blythe, David and Tom McReynolds. "Programming with OpenGL: Advanced Rendering". SIGGRAPH '96: 51-57.
- [3] Filkorn, Roman and Marek Kocan. "Simulation of Human Body Kinematics". 2000. On-Line. Internet. 28 Mar. 2001. Available: <http://www.cg.tuwien.ac.at/studentwork/CESCG-2000/RFilkorn/>
- [4] "glBlendFunc". n. pag. On-line. Internet. 26 Mar. 2001. Available: <http://www.sun.com/software/graphics/OpenGL/manpages/glBlendFunc.html>
- [5] Johns, Mike. "mHockey". n. pag. On-Line. Internet. 1 Mar. 2001. Available: <http://www.linuxgames.com/mhockey/>
- [6] Kilgard, Mark. "The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3". n. pag. On-line. Internet. 28 Feb. 2001. Available:

- http://reality.sgi.com/mjk_asd/spec3/spec3.html
- [7] Kilgard, Mark. "OpenGL-based Real-Time Shadows". N. pag. On-line. Internet. 22 Mar. 2001. Available: http://reality.sgi.com/mjk_asd/tips/rtsl/
- [8] Kuo, Art. "Passive Walking". n. pag. Internet. 27 Feb. 2001. Available: http://www-personal.engin.umich.edu/~artkuo/Passive_Walk/passive_walking.html
- [9] "Matrix and Quaternions FAQ, The". n. pag. On-line. Internet. 27 Feb. 2001. Available: <http://skal.planet-d.net/demo/matrixfaq.htm#Q41>
- [10] Nilsen, Gregory S. "Artificial Intelligence In Sports: A Study Upon American Football". Hiram College Department of Computer Science Website. On-line. Internet. 1 Apr. 2001. Available: http://cs.hiram.edu/~nillsengs/AI_ProjPaper.PDF
- [11] "Pre-Game Overview". n. pag. On-line. Internet. 19 Feb. 2001. Available: <http://www.football.com/rulesabc/pre-game.shtml>
- [12] Robins, Nate. "Nate Robins – OpenGL – Tutors". n. pag. On-line. Internet. 28 Mar. 2001. Available: <http://www.xmission.com/~nate/tutors.html>
- [13] "Welcome to the OpenGL On-Line Reference!". n. pag. Internet. 2 Mar. 2001. Available: http://www.hp.com/workstations/support/documentation/manuals/user_guides/graphics/opengl/OpenGL.html