Physical and Graphical Golf Simulation

Jeff Sturgis - 261437

August 8, 2003

COMP 4905
Honours Project

Supervised By:

Dr. Mark Lanthier
School of Computer Science
Carleton University

Abstract

This project contains an implementation of a Hole Creator, game simulation, and 3D engine. It was inspired by years of ideas directed towards a golf game better than any other. Three very important elements for any golf game are physics, usability, and graphics. The Hole Creator allows anybody to create golf holes quickly and accurately. Multiple golf holes will be important to allow for a large variety of testing. Also, the ability to create holes is appealing to users who want to create their own holes and play them.

This project also includes 2D and 3D game engines. The currently used engine is the 2D engine. This engine shows the hole as a top view, allowing the user to see the ball location and height as it flies. The 3D engine is a modified surface graphing project, which plots surface functions. It plots the terrain in 3D and will be used in the future as the primary game view. Major improvements to the 3D engine are still under way. They will allow faster and cleaner rendering. Until these improvements are made the 3D engine takes too long to show the holes and thus greatly increases the testing time.

The physics of the four main forces acting on a ball whilst flying through air are gravity, velocity, drag, and magnus. These forces are accounted for as well as the forces behind the ball landing and rolling. By using surface functions, these calculations are handled with very high precision.

Acknowledgments

Sun Microsystems provided me with Java(tm) as well as their exhaustive API.

Java 2D O'Reilly by Jonathan Knudsen is an excellent book that helped me create the Hole Creator. The Hole Creator is the foundation for all the graphics in the game.

Physics for Game Developers O'Reilly by David M. Bourg explains the forces that act on a golf ball in great detail.

Yanto Suryono released an open source project entitled Surface Plotter. This project is the foundation for the 3D engine. I would like to thank him for his very clean understandable code and praise him for releasing it open source.

James Stewart wrote the book Multivariable Calculus, and answered my email about solving for an intersection between a surface and vector function.

Linux utilities such as Dia for diagram editing, Gimp for graphics editing, and LaTeXin which this document was written.

Craig Butfoy released an excellent comprehensive guide to golf. In this he includes terminology, rules of game play, and types of golf shots. This document can be found at `http://www.surreyladiesgolf.co.uk/CRAIGS%TIPS.pdf` and is a great reference for any golfer.

Thanks to:

Dr. Mark Lanthier for taking on this project and guiding me through it.

Cheryl Young for proofreading this document as well as her patience. She also helped motivate me to do this project.

Crystal Sirard for proofreading.

Dave Kretz who got me started on the Hole Creator and helped mold me into a Java guru. I would especially like to thank Dave for the time he saved me by providing me lines of code that allow applets to access files within a jar. With this information the game is now able to load holes and images from a browser.

My mother for lending me her digital camera that saved me a lot of time on graphic design.

Co-worker Keean Robertson at Greyhawk Golf Club for making me aware of the forces imposed on a golf ball and answering many golf related questions.

Contents

List of Figures

List of Tables

1 Introduction

In the beginning, golf games such as PC-Golf that ran on a 286, allowed the player to input their club and hit the ball by striking enter. Golf games have come a long way since that time and now allow for input such as power, error, and spin. Wind is now also taken into consideration, as well as complex terrain. Some modern golf games even allow users to play different types of shots such as a low punch shot, or high flop shot.

The purpose of this project is to create a foundation for a future full-fledged online golf game. The design of the model is very important to this project because it is the model, which will determine the running time of loading holes and calculating shots. This project will contain a Hole Creator, a four-hole game, and a preliminary 3D engine. The game will be fully functional with the exception of the database and everything that relies on the database.

1.1 Game Advantages

This game will eventually include everything other golf games offer and much more. For example, different shots are carried out through adjustments of the stance, grip, and ball position. This game will contain a high level of detail, which will allow the experienced player to customize their shots with great accuracy. The inexperienced player may use pre-programmed shots and learn the proper stance and grip. This game is designed not only to be fun, but as an accurate golf simulation which will teach people how to play golf, how different parameters affect every shot, and allow people to practice golf without leaving their home.

Another advantage to this game will be its ability to store information on a server. Golf realism can be further extended in many ways. For example, in other games when you lose a ball it does not matter because you have infinitely many balls. In this game when you lose a ball, a ball can be subtracted from the database. As in real golf, it will be possible for a player to run out of balls. Though the game will most likely allow unlimited access to inferior balls, players can still suffer the disappointment as found in real golf, when losing a brand new top of the line ball. The database will also allow players to get golf clubs. By taking these clubs to the driving range, the players can not only get a sense of how far the club goes, but the average yardage for a club can be stored in the database. Club selection can then be used based on the average distance of the club.

This game will be divided into two modes. One mode of the game will allow people to practice using their equipment and play online with friends. This mode will be low security and may allow cheating. The database will store minimal information in this mode, such as the final score of a round and possibly a few terrific golf shots that they made. The second mode is the tournament mode. Security in this mode will be strictly enforced. Cheating will not be permitted. Some tournaments may cost money to enter. However, each shot in tournament mode will be saved for logging purposes as well as giving the user something back in return. The player will be able to reload any hole and view how they played it. As a direct consequence, the player need not save any great shots that they made because they will be saved automatically.

The currency in this game will be a point system similar to the Air Miles, or

Petro Points. Points will allow the user to buy new clubs and other golf equipment. They will also be able to trade in their old clubs for points, which they can put towards a new set. In order to get users playing this game, points will be given away for free. If the game generates a lot of traffic, servers will need to be upgraded and the game may start charging people for points. Points will be awarded to new accounts in a controlled way. This way shall not upset existing members and minimize their motivation to create a new account. One way to implement this is to change the new account bonus regularly and when doing so, give all existing members every new account bonus automatically.

With the help of sponsors, this web site will be able to remain free. However, since the oldest accounts will be awarded the most points and other benefits, new users who wish to catch up will be able to buy their way into top of the line status. For example, before a user can play in a tournament they must play twenty rounds of golf. After twenty rounds of golf, the player will be awarded a handicap. If they wish to enter tournaments without playing twenty rounds then they will have the option of buying a handicap. Also, if they do not have enough points to get the best set of clubs and they want the best set, they could buy additional points and upgrade their clubs.

Prizes will be awarded to those who enter tournaments. The people who play the best will be awarded the most points, or possibly the option of selecting their prize from a list. The point system works very well in this situation because many golf tournaments award players gift certificates to the pro shop. Essentially, this is the same as awarding points, which can be used in the game's online pro shop.

Tournaments will range in difficulty as well as the type of game. Only users with lower handicaps may register for more difficult tournaments. The more difficult the tournaments are, the better the prizes will be for the winners. This will give users incentive to become better. There will be many different types of tournaments such as team scramble or match play. Some tournaments may last several rounds and eliminate players each round.

This game will allow players to create golf holes. Points will be awarded to players who create good golf holes. If users are looking to obtain more points without paying, it will be possible to sign them up for maintenance. Players may be assigned jobs such as, cutting fairways, setting the pin, emptying garbages, or filling water coolers. They will be awarded points like a pay cheque. This will ensure that the game can stay free to people who really do not want to pay, but would like to maintain their status with the best clubs. Another possibility to obtain free points is by creating a caddy. Caddy creation would allow users to submit audio clips that their caddy would say. For example, if the ball landed in the water, the caddy might say ”tough break” or ”looks wet to me”.

A free online game controlled by a database will give this game more originality, realism, and user control than any other game.

2 The Model

The model for this project is broken down into the mathematical and physical components. The most complex areas for the model are the surface and golf shot implementations.

2.1 Surfaces

In computer graphics, to render properly you must have points that can be plotted. Many terrain generators maintain an object mesh for each object in the game. This requires memory to keep track of plottable points and a level of detail algorithm to reduce superfluous calculations.

2.1.1 Height Function

The golf holes rely on a different structure to represent the surface terrain. Though, it is impossible to avoid a level of detail algorithm for calculation reduction, it is possible to reduce the memory storage for the object meshes. This is an advantage for an online game that is downloaded by many people, because it will reduce the amount of data transferred.

The terrain on a Surface is represented by a two-variable height function in the form

$$z = f(x,y) : x > 0, y > 0, x, y \in \mathbb{R} \tag{1}$$

The advantage to storing terrain in this way is that there is no need for an object mesh. Points can be calculated with extremely high precision. This function of two variables is very useful but only maintains the height of the object.

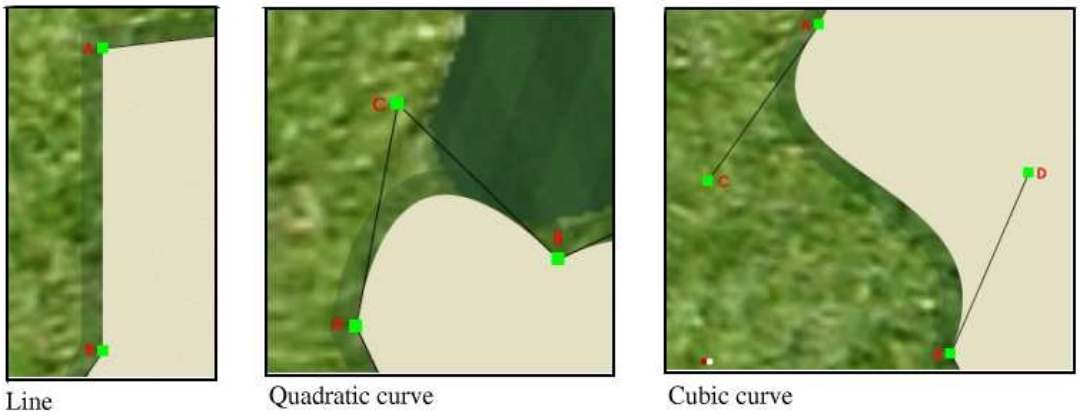

Figure 1: Lines and Curves with their Corresponding Point Representation

2.1.2 Surface Border

To maintain the border of the Surface, points are used. The points are somewhat minimized though by using curves. A line requires two points, a quadratic curve requires three points, and a cubic curve requires four points. Figure 1 shows the three different types of lines used to create the Surface border. By connecting these curves or lines into a shape, the outline of the Surface can be drawn. The points not only keep track of the shape of the Surface but they also keep track of the Surface location on the map of the hole. [5]

2.2 The Magnitudes for a Flying Golf Ball

The forces exerted on the ball whilst flying through the air are the velocity of the ball, acceleration due to gravity, drag, and the magnus force.

2.2.1 Initial Velocity

The force vector behind the initial velocity of the golf ball is proportional to the product of the mass of the ball and the acceleration of the club head. The direction is perpendicular to the face of the club (assuming the ball was hit properly). From Newton's second law of motion:

$$\vec{F} = m \times a \tag{2}$$

$$\vec{F} = m \times \frac{dv}{dt} \tag{3}$$

$$\vec{F} \times dt = m \times dv \tag{4}$$

$$\int_{t_1}^{t_2} \vec{F} dt = \int_{v_1}^{v_2} m\, dv \tag{5}$$

$\vec{F}$ is the force vector, m is the mass of the ball, and a is the acceleration of the club. Time t_1 can represent the beginning of the shot, therefore we have $t_1 = 0$. We get,

$$\int_{0}^{t_2} \vec{F} dt = \int_{v_1}^{v_2} m\, dv \tag{6}$$

$$\vec{F} \times t_2 = m \times v_2 - m \times v_1 \tag{7}$$

The clubs acceleration through the ball is not important. What is important is the instant of acceleration when the club meets the ball. This will give us our initial velocity. However, there is an easier way to calculate precise initial velocity. For this game, backward substitution into the position function was used to find appropriate initial velocities. That is, by inserting different initial velocities into the position equation (taking into consideration all forces) and use the initial velocity corresponding to how far the ball should to go. For example, the restriction in which the ball may not fly any longer than is 320 yards off the tee on flat elevation and with no wind. So if an initial velocity is used that allows the driver to go 320

yards, this is the correct initial velocity. Consequently, by using that same initial velocity for every club, the yardage decreases for each club giving the correct yardage.

According to Newton's first law of motion, a body will remain at rest, or continue in rectilinear motion unless an external force is imposed on it. Assuming no force is imposed on the golf ball, it should continue in a rectilinear direction perpendicular to the face of the club. However, other forces on the ball such as gravity will alter the balls position.

2.2.2 Gravity

Let g represent the accelerative force due to gravity which is approximately $9.8\,\mathrm{m}/\mathrm{s}^2$ towards the earth. Let $\vec{F}$ be the acceleration vector with respect to gravity, $\vec{v(t)}$ be the velocity function, and $\vec{r(t)}$ be the position function with respect to gravity. Since velocity is the derivative of position with respect to time, and acceleration is the derivative of velocity with respect to time, it follows that velocity is the integral of acceleration with respect to time and position is the integral of velocity with respect to time. Let $\vec{i}$ be the unit vector along the horizontal and $\vec{j}$ be the unit vector along the vertical.

We get,

$$\vec{F} = \mathrm{m} \times a \tag{8}$$

where $a = -g \times \vec{j}$, then

$$\vec{v(t)} = -g \times \mathrm{m} \times \vec{j} + \mathbf{C} \tag{9}$$

and $\mathbf{C}$ is equal to our initial velocity vector $\vec{v_0}$. Therefore,

$$\vec{r'(t)} = \vec{v(t)} = -g\mathrm{m}\vec{j} \times t + \vec{v_0} \tag{10}$$

Thus,

$$\vec{r(t)} = -g\mathrm{m}\vec{j} \times t^2 + \vec{v_0} \times t \tag{11}$$

This means that the position of the ball will follow a parabolic curve with respect to time. Since gravity is a force towards the centre of the earth $(-g)$, the resulting curve is an upside down parabola which is no surprise.

By applying this force to the initial velocity $v_0 = |\vec{v_0}|$, and using

$$\vec{v_0} = v_0 cos\alpha \times \vec{i} + v_0 sin\alpha \times \vec{j} \tag{12}$$

we can arrive at a distance function $d(t)$ and height function $h(t)$ such that

$$d(t) = v_0 \cos\alpha \times \vec{i} \tag{13}$$

$$h(t) = [v_0 \times sin(\alpha) \times t - \frac{9.8}{2} \times t^2] \times \vec{j} \tag{14}$$

where α is the loft of the club. [6]

2.2.3 Drag

Newton's third law states that for every action, there is an equal and opposing reaction. An opposing reaction to the ball flying through the air is the air resistance. As the ball pushes through the air, the air pushes back slowing down the ball. The effect of drag on the golf ball will change the trajectory of the ball such that it no longer follows the path of a negated parabola, but rather a skewed parabola as shown in Figure 2.

If $\vec{F}$ is the force vector acting on the ball with respect to initial velocity and gravity, then the force of drag is

$$-c \times \vec{F}, \text{ for some scalar } c \text{ such that } 0 < c < 1 \tag{15}$$

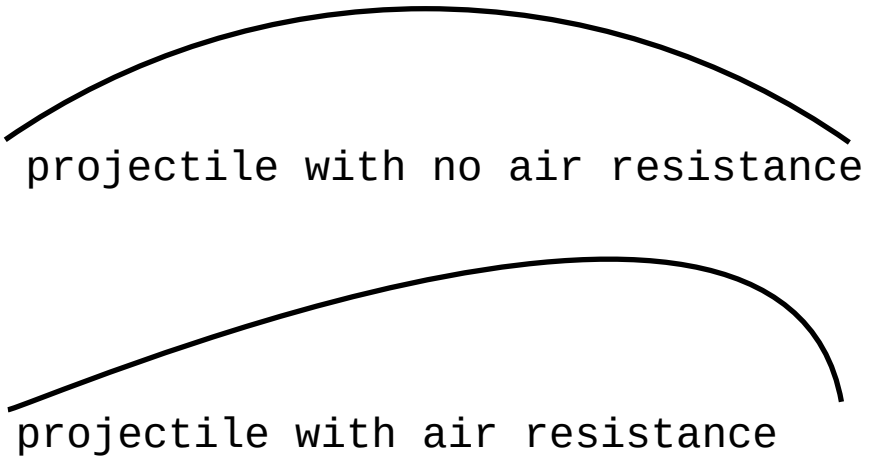

Figure 2: Trajectory of Ball With Drag

Notice that since drag is a force against the projectile, wind plays a major factor. The affect of drag is heightened when the direction of wind is against the ball. As well the affect of drag is reduced when the wind is in the same direction of the ball. Consequently, the force of wind imposed on the ball can be implemented via the force of drag. [4]

2.2.4 Magnus

A not so obvious force imposed on the ball is the magnus force. When the golf ball is stricken, spin is placed on the ball. If the ball is hit perfectly square and the path of the club head travels perpendicular to the clubface, the imposed spin is backspin. It is possible to put topspin on the ball, however, this would require hitting the upper half of the ball. A direct consequence of this would be the ball hitting the ground soon after the club has made contact. For this reason, there is no topspin in the game. If the club is square but makes contact in an out to in motion (for a right-handed player) the imposed spin is back right. If the motion of

the club through the ball is in to out, the spin is back left. By inverting the club, the forces imposed on the ball remains the same for a left-handed player. However, the swing patterns are inverted. That is, in to out for a right-handed player is out to in for a left-handed player. A fade for a right-handed player is a draw for a left-handed player. The same inversion occurs with a right-handed draw.[2]

Figure 3 illustrates the three different ways to hit the club square and impose a different spin on the ball. The first case is when the ball is struck square and the club comes through the ball straight. Assuming they hit the ball in the middle of the face of the club, the imposed spin will be backspin creating a lifting effect on the ball. The second case is when the club is square to the ball but comes inward towards the body in the follow through. The imposed spin for a right-handed player would be back right. The result will send the ball to the left initially and curve the ball to the right due to the magnus effect. The last case is opposite to the second case. If the club is square when hitting the ball but heading away from the body in the follow through, then the resulting spin is back left and will start the ball out to the right curving it to the left due to magnus. Figure 4 illustrates the effect of backspin on the wind flow around the dimpled golf ball. As the air flows over the top of the ball, the direction of the spin is with the airflow. This speeds up the airflow over the top of the ball. However, on the bottom of the ball the spin is against the airflow that slows down the air under the ball. The resulting airflow pushes the ball up. Though it is hard to see in the diagram how this could be so, think of the ball like a boat propeller and the red arrows to be the direction which the propeller pushes the water. The boat would go in the opposite direction of the force on the water. That is why the ball goes in the opposite direction of the red

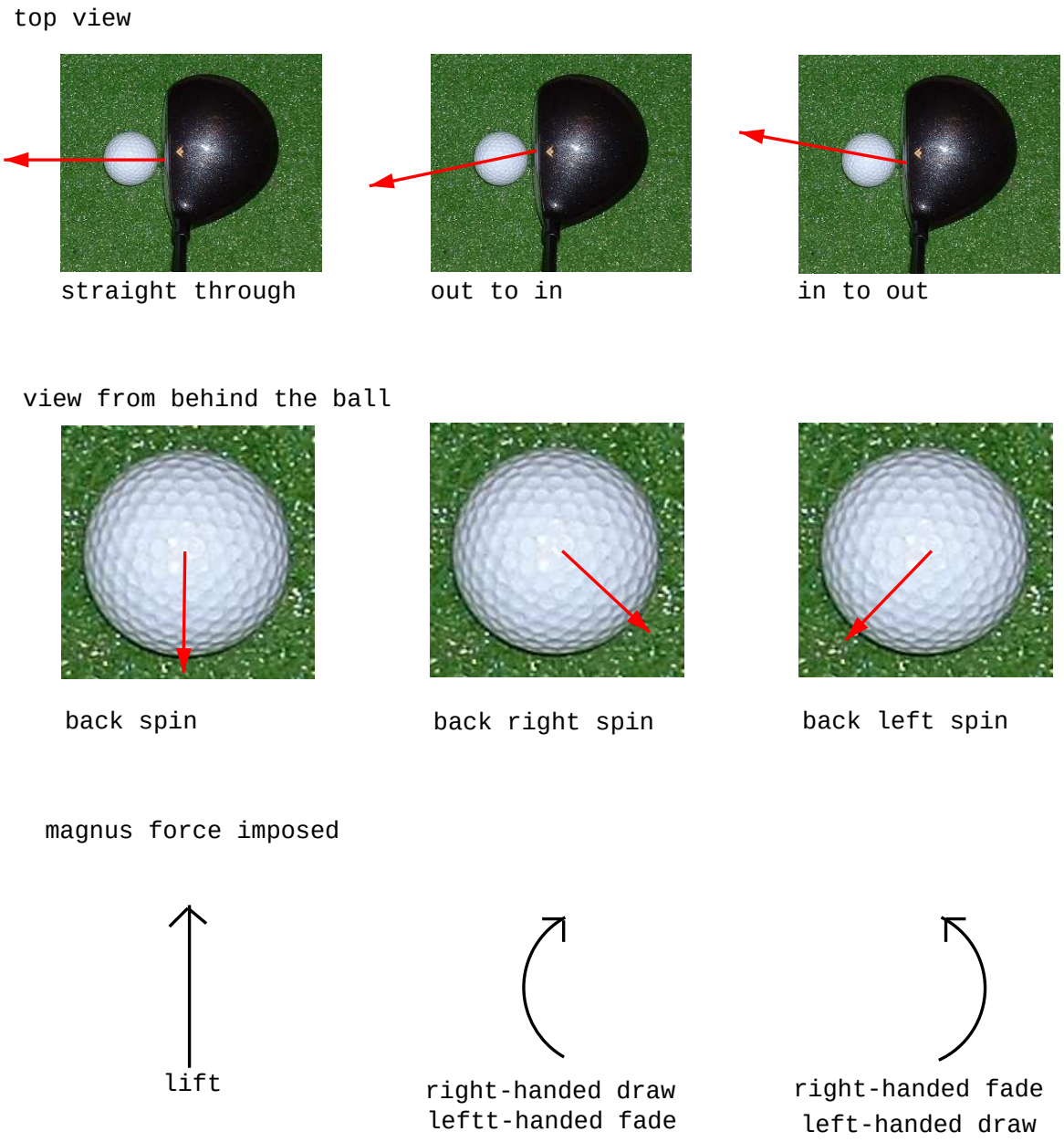

Figure 3: Spin Imposed Due to Club Contact

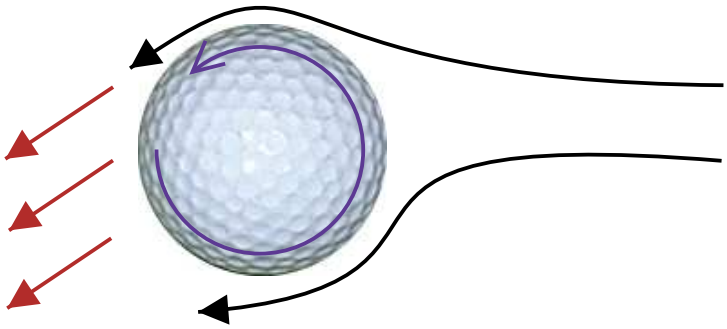

Figure 4: Motion of Ball Through the Air

arrows.[4]

A perfect fade or draw will land at the target.[2] This is how the game implements draw and fade. The amount of right or left spin imposed on the ball will determine how much the ball fades or draws. To ensure that these shots are not trivial, the player must aim the target precisely where the ball will land. Failure to do so will result in an imperfect shot. If the player aims the target short of where the ball will land, the spin on the ball will be too much and the ball will end up right of the target on a fade or left of the target on a draw. On the other hand, if the target is placed too far for the shot, the ball will land before it is done curving and the result will be left on a fade and right on a draw. This is assuming the player is right-handed.

The amount of backspin imposed on the ball will also determine how far the ball goes. The more backspin the further the ball will go. In reality it is possible to inflict too much spin on the ball, however modern clubs are designed not to do this so the game will not handle such a scenario. This game implements the mag-

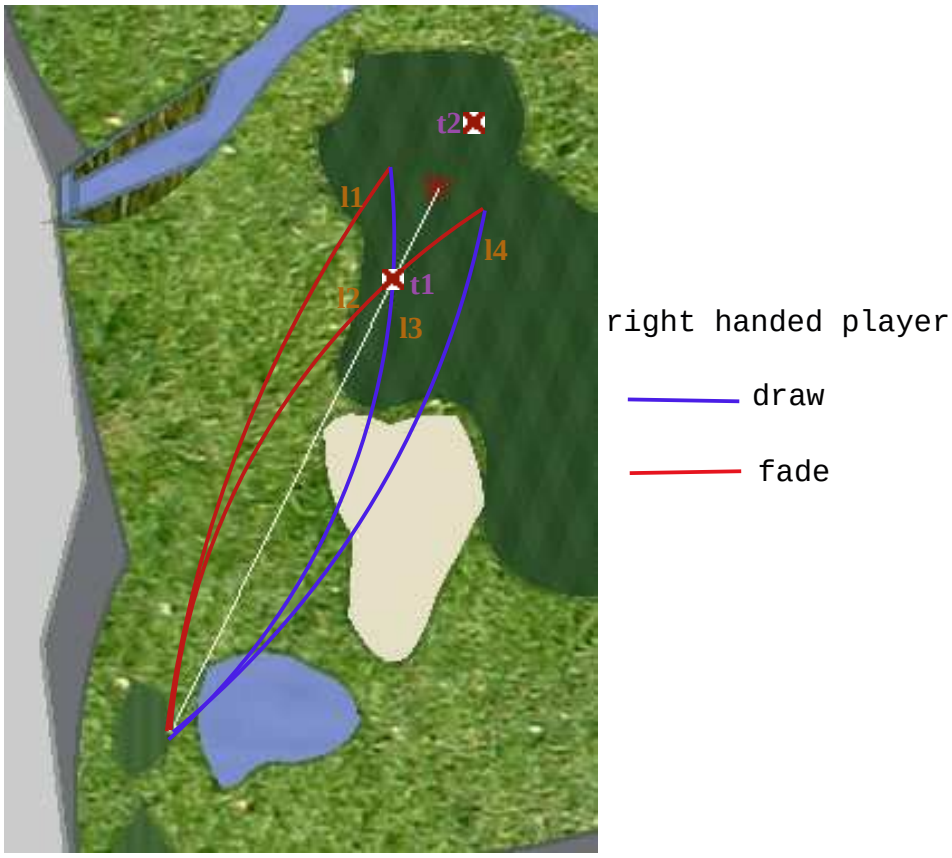

Figure 5: Implementation of Fade and Draw

nus force of backspin as a scalar vector upwards. The implementation of sidespin is shown in figure 5. This figure shows four cases of golf shots, two fades and two draws. The shots which surpass the target line (l2 and l3) are caused by the target being chosen too closely to the origin (t1). The cases where the ball does not curve to the target line (l1 and l4) are a result from the target being selected beyond the distance of where the ball will land(t2). The white line to the target represents the correct distance for the shot. Had this been the selected target, the ball would have curved and landed exactly at that point.

2.3 Direction for a Flying Ball

An easy way to visualize the position of the golf ball is to assume time as the x-axis and height as the y-axis. Thus, the ball position is merely a function of height over time $h(t)$. This is essentially what is happening, however, the position of the ball on the map of the hole must change with respect to x and y. If Θ is the angle made between the ball and the target, then the ball will change in the x direction by the function

$$v_0 \times cos(\Theta) \times t \times drag \tag{16}$$

and the ball will change in the y direction by the function

$$v_0 \times sin(\Theta) \times t \times drag \tag{17}$$

Figure 6 shows the triangle used to calculate the direction of the ball. By subtracting the origin point from the destination point, the values for x and y can be obtained. With these values, the angle Θ can be found and the ratios $sin\Theta$ and $cos\Theta$ can be used to project the x position and y position with respect to the horizontal $\vec{i}$.

2.4 Ball Landing and Bounce Calculation

Let $r(t) =< f(t), g(t), h(t) >$ be the vector function of the ball. The vector function will intersect with the terrain $z = k(x,y)$ at time t_1 because

$$\lim_{h(t) \to k(f(t),g(t))} r(t) = t_1, t > 0 \tag{18}$$

That is, as the height position in the vector function approaches the height of the terrain they will eventually arrive at an intersection which will occur at time t_1. The point of intersection then is $P(f(t_1), g(t_1), h(t_1))$. To find the tangent

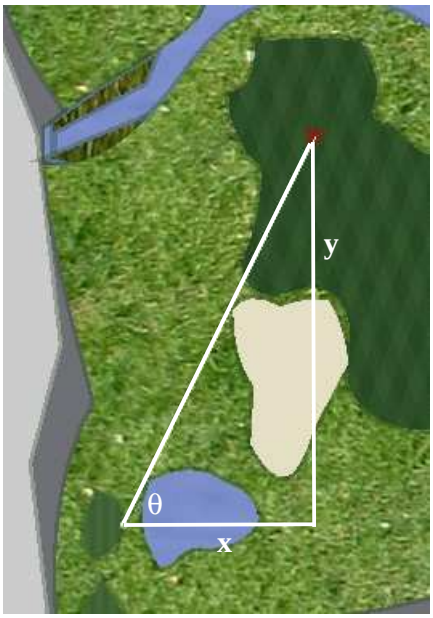

Figure 6: Triangulation Between Ball and Target

vector to the vector function $r(t) =< f(t), g(t), h(t) >$ at time t_1 we use $r'(t) =< f'(t), g'(t), h'(t) >$ and substitute in t_1. Thus, our tangent vector is

$$< f'(t_1), g'(t_1), h'(t_1) > . \tag{19}$$

Given the point of intersection of the terrain to be $P(x_1, y_1, z_1)$ we can use the formula

$$z - z_1 = f_x(x_1, y_1)(x - x_1) + f_y(x_1, y_1)(y - y_1) \tag{20}$$

to find our tangent plane, where f_x is the partial x derivative of the surface function, or rather the derivative of the surface function with respect to x, and f_y is the derivative of the surface function with respect to y. When arranging the tangent plane equation into the form

$$Ax + By + Cz + D = 0 \tag{21}$$

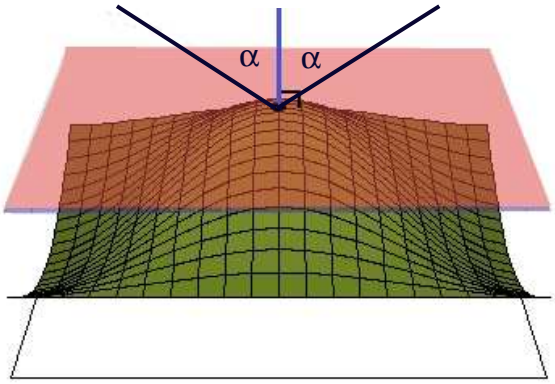

Figure 7: Normal Vector

we can obtain the normal vector simply by using the vector $< A, B, C >$. Thus,

$$f_x(x_1, y_1)x + f_y(x_1, y_1) - z - f_x(x_1, y_1)x_1 - f_y(x_1, y_1)y_1 + z_1 = 0 \tag{22}$$

and we get

$$< f_x(x_1, y_1), f_y(x_1, y_1), -1 > \tag{23}$$

as our tangent plane.[6] Figure 7 shows that the angle of incidence to the normal is equal to the angle of reflection. The tangent plane to the surface will likely be on an angle, so in order to calculate the angle of incidence you simply calculate the angle (α) between the tangent vector of the golf shot at the point of intersection with the surface to the normal vector.

2.4.1 Absorption

When the ball strikes the ground on a hard terrain, the ball will have a greater bounce than if it bounced off a soft terrain. This reaction is implemented by $absorption$ such that $0 < absorption < 1$. As $absorption \to 1$ the ball will bounce

with a greater magnitude. As $absorption \rightarrow 0$, the ball will bounce with a less magnitude.

2.5 Putting and Rolling

2.5.1 Directional Surface Vector

To find the direction of the surface $z = f(x,y)$ at any given point, we take the partial derivatives of the surface function. For instance, if the given values for (x,y) are (x_1,y_1), then the partial derivatives are

$$f_x(x_1,y_1) = \frac{d}{dx} f(x,y), \text{ and substitute in } (x_1,y_1) \tag{24}$$

$$f_y(x_1,y_1) = \frac{d}{dy} f(x,y), \text{ and substitute in } (x_1,y_1) \tag{25}$$

These partial derivatives give the rates of change of z in the x- and y-directions. Let us denote the gradient vector $\vec{\Delta} f$ to be $< f_x, f_y >$. The gradient vector represents the greatest rate of change for the terrain. That is, if the ball is hit along the gradient it will not stray from it's path. However, any putt which is not hit along the gradient vector will tend to break towards the gradient. Keep in mind this is only for a small change in value, an entire putt will break many times along many gradient vectors.[6]

The slope of the terrain can be calculated as

$$slope = |\vec{\Delta} f| \tag{26}$$

Recall from equation (22) that the normal vector at $P(x_1,y_1,z_1)$ is

$$< f_x(x_1,y_1), f_y(x_1,y_1), -1 > \tag{27}$$

Let S_n represent the slope of the normal vector. The rise of the normal vector is 1 and the run is $\sqrt{f_x(x_1,y_1)^2 + f_y(x_1,y_1)^2}$, thus the slope of the normal vector is:

$$S_n = \frac{1}{\sqrt{f_x(x_1,y_1)^2 + f_y(x_1,y_1)^2}} \tag{28}$$

$$S_n = \frac{1}{|\vec{\Delta} f|} \tag{29}$$

However, since the tangent plane is perpendicular the normal vector, the slope is

$$slope = \frac{1}{S_n} \tag{30}$$

$$slope = \frac{1}{\frac{1}{|\vec{\Delta} f|}} \tag{31}$$

$$slope = |\vec{\Delta} f| \tag{32}$$

As the ball rolls along the ground, it gradually slows down. When the force behind the ball is less than the force of gravity, the ball will remain stationary. If the slope of the tangent plane to the terrain at P is an incline, the ball will slow quicker. If the slope is a decline, the ball will slow down less quickly. Thus, if the ball is rolling along the gradient, then the ball will be rolling on a decline. Similarly, if the ball is rolling against the gradient it will be on an incline.

2.5.2 Friction

Let σ denote the slope of the tangent plane which will cause the ball to travel at a constant speed. Thus, if the angle of the tangent plane is steeper than σ, the ball will accelerate down hill. Another force involved in surface rolling is friction. If the ground imposes a lot of friction on the ball, such as tall grass, then σ will be increased (steeper). If the ground does not impose much friction, σ is decreased (flattened). The friction of a Surface is implemented as a value such

that $0 < friction < 1$. As $friction \to 1$, the Surface will slow the ball down more and as $friction \to 0$, the ball will slow down less.

3 Design and Implementation

The objects in this game that are instantiated will eventually be loaded from a database. For this reason, equipment objects contain a unique ID variable so that the player can obtain an object by sending a request to the server with the object's ID.

Until the database is operational, the objects will be loaded from selection classes. For example, GolfClub objects are loaded from the ClubSelection class, GolfBall objects from the BallSelection class, etc.

3.1 Equipment Objects

This game implements two main equipment classes, GolfClub and GolfBall. The GolfClub class maintains the characteristics of the club such as ID, name, length of shaft, lie of the clubface, loft of the clubface, and distance which is a variable represented by the magnitude of the force vector imposed on a perfect golf shot. The force vector would then be $distance \times cos\alpha \times \vec{i}$, and $distance \times sin\alpha \times \vec{i}$, where α represents the loft of the club.

The GolfBall class maintains characteristics such as ID, name, as well as variables representing how well the ball performs. The variables spin, distance, and control are such that $0 < \ distance, \ spin, \ control \ < 1$. This will allow the ability to make different golf balls each having different individual performances. For example, the game will be able to make a distance ball, which will go further than a spin ball, or a spin ball that will spin better than a distance ball.

3.2 Game Objects

Viewing the file game.html as an applet runs the game. The GolfGamePanel contains a HolePanel, SwingPanel, ClubPanel, and LiePanel. The HolePanel loads a golf hole through a class called ShapeCreator. ShapeCreator is discussed further in section 4.1.4. The HolePanel graphically maintains the position of the ball as well as the physical target. The HolePanel zooms in on the Surface which currently contains the ball, unless the ball is on the green. If the ball is on the green, the view will be zoomed in further to ensure a good view of the hole. The ClubPanel is used to select the club as well as the speed of the swing. The type of shot selection which is also on the ClubPanel will be implemented later to allow the user to select different shots. The LiePanel uses the Lie class to get the current ground position of the ball and display the lie. For example, the LiePanel will show a picture of the ball on the fairway if the lie of the ball resides on the fairway.

The purpose of the SwingPanel is to get the user input for the golf shot. When the player is swinging any club other than a putter, it will display a curved swing path. When the user clicks on the swing button, the swing timer starts and draws the position of the club head based on time. The putt timer would start if the player is putting and it would draw the position of the putter based on time. When the player selects the power and accuracy of their shot, a golf shot object is created with the input of the golfer, golf club, ball, power, error, wind, spin, origin, and target. The golf game class keeps track of which player is playing so that in the future, multiplayer functionality can be added.

3.3 Graphics Objects

Thanks to the code of Yanto Suryono[3], the 3D engine is under way and without much effort, already draws holes in 3D. When the SurfacePlotter class is run, the SurfaceFrame parses a surface function from a string. It then creates an array of SurfaceVertex objects by calculating the heights of points in the surface function using x- and y- regions. The SurfaceCanvas paint method is called which uses a projector to project all the points in the array of surface vertices and draws it.

In order for the SurfacePlotter to plot golf holes, the way in which the surface vertices are obtained has been overwritten. The SurfaceFrame now calculates which Surface is at the point before finding the height at that point. Surface vertices also keep track of the colour of the Surface so that when drawing, it need not look it up. The paint method is pretty much the same except that it obtains the colour for the SurfaceVertex instead of using a function of height as it did before.

3.4 Physics Objects

A VectorFunction is like a vector except that its quantities are functions rather than constants. In this game, the functions within the VectorFunction class are based on time. For this reason, VectorFunction contains a getPoint method which takes in time and calculates their x-, y-, and z-positions. A NormalShot is a regular golf shot which takes into account initial velocity, gravity, drag, and magnus forces. The sidespin for magnus force is represented in the Spin class, and the backspin is maintained in NormalShot. When the swing input is gathered a GolfShot is created which in turn creates a CalculatedShot. The job of the CalculatedShot is to keep the ball moving as long as it is in motion. It starts by calculating the

angles from the origin of the ball to the target. It then creates a NormalShot. The CalculatedShot keeps track of the height of the ball at every interval and checks to see whether the ball has come in contact with the ground. When the ball makes contact with the ground, the CalculatedShot computes the angle of incidence and reflection. As well, the CalculatedShot computes the forces between the ball and the ground and decides how much absorption the ground has on the ball. If the force of the ball is great enough to bounce, it will create another NormalShot based on new values. Eventually, the ball's force will not be great enough to bounce in which time it will begin to roll. The CalculatedShot then creates a Roll that will compute the forces of the ball with respect to ground friction and gravity.

3.5 Hole Creator Objects

Creating a hole begins by making a new hole and inputting the name and par of the hole. The surfaces can then be drawn using different curves. Subsurfaces can be added to a Surface which will have their own image texture, absorption, height function, and name. The undo option so far allows the user to undo lines which they created by accident. This is kept track of in a Job stack such that whenever a new line is created, a corresponding Job is created and pushed on the stack.

The GolfHole is made up of Surface object, and keeps track of the position of the pin, and tee decks. Surface uses a string to represent the height function. Once this string is parsed, the height at any point can be readily obtained. The structure of the hole is aVector tree starting with the out of bounds Surface. Each Surface contains a Vector of subsurfaces, so that each point of a Surface may exist only if it is contained in it's parent's Surface.

Once all the Surface objects are created the user can set the pin, and tees. This feature is realistic because pin and tee placements on a golf course are changed on a daily basis. By changing the position of the hole on the green, (resetting the pin) the hole can play very differently. The pin could be placed at the back of the green allowing for an extra club to be used, or at the front, where one less club would be used. Also, depending on the difficulty of the green, the hole could be placed on a plateau or along a break. Tee decks do not have as much affect on the play of the hole. However, it does add realism to the game by allowing them to be reset easily.

3.6 Object Interaction

3.6.1 Game Object Interaction

If p1 and p2 represent the two points, ball position and target respectively, then in the case of a perfect shot with no sidespin, the ball should follow the path along a line created by p1 and p2 towards the direction of p2. The target represents the line of the ball, not how far the ball is hit. For a straight shot, the ball will follow a more accurate path if the target is further away than the hitting distance of the club. However, for a shot with sidespin, draw or fade, the ball will follow a more accurate path if the target is placed where the ball should land. In the case of a right-handed fade, if the target is short of where the ball lands, the ball will have too much spin and end up right of the target. If the target exceeds the distance of the ball, the ball will not have enough spin and fall left of the target. This is the same for a left-handed draw and opposite for a right-handed draw or left-handed fade.

The backspin on the ball creates lift. As a ball spins in the air, the dimples create turbulent airflow that moves the ball up or down, for backspin and topspin respectively.

The shot begins by interpreting which type of shot was made. In the case of a putt, only the Roll need be calculated. Other shots create a NormalShot which is a ball path through the air. The input for the shot that is gathered from the SwingPanel is the sidespin on the ball, force, and error. The input from the ClubPanel will determine the type of shot and which club is used. The origin and target are obtained from the HolePanel. With all this information, a CalculatedShot is created. In the case of a NormalShot, VectorFunctions for the NormalShot and Spin are created. As the ball flies through the air the CalculatedShot checks whether the ball has fallen below the terrain. When this happens the bounce is calculated, force and direction are updated, and a new NormalShot is created with the updated values. The ball will continue to bounce off the ground until its vertical force vector is less than the force of gravity. At this point a Roll shot begins. The CalculatedShot continues to update time and get the position of the ball with respect to ground slope. Eventually, the horizontal force will subside and the ball stops.

By clicking on the SwingPanel, the shot is reset. The player should then choose their next club, spin, and go again. If the player selects the putter, the putt shot is used as default. The SwingPanel also changes to a putting stroke rather than a swing. Figure 8 shows the class diagram for the golf game.

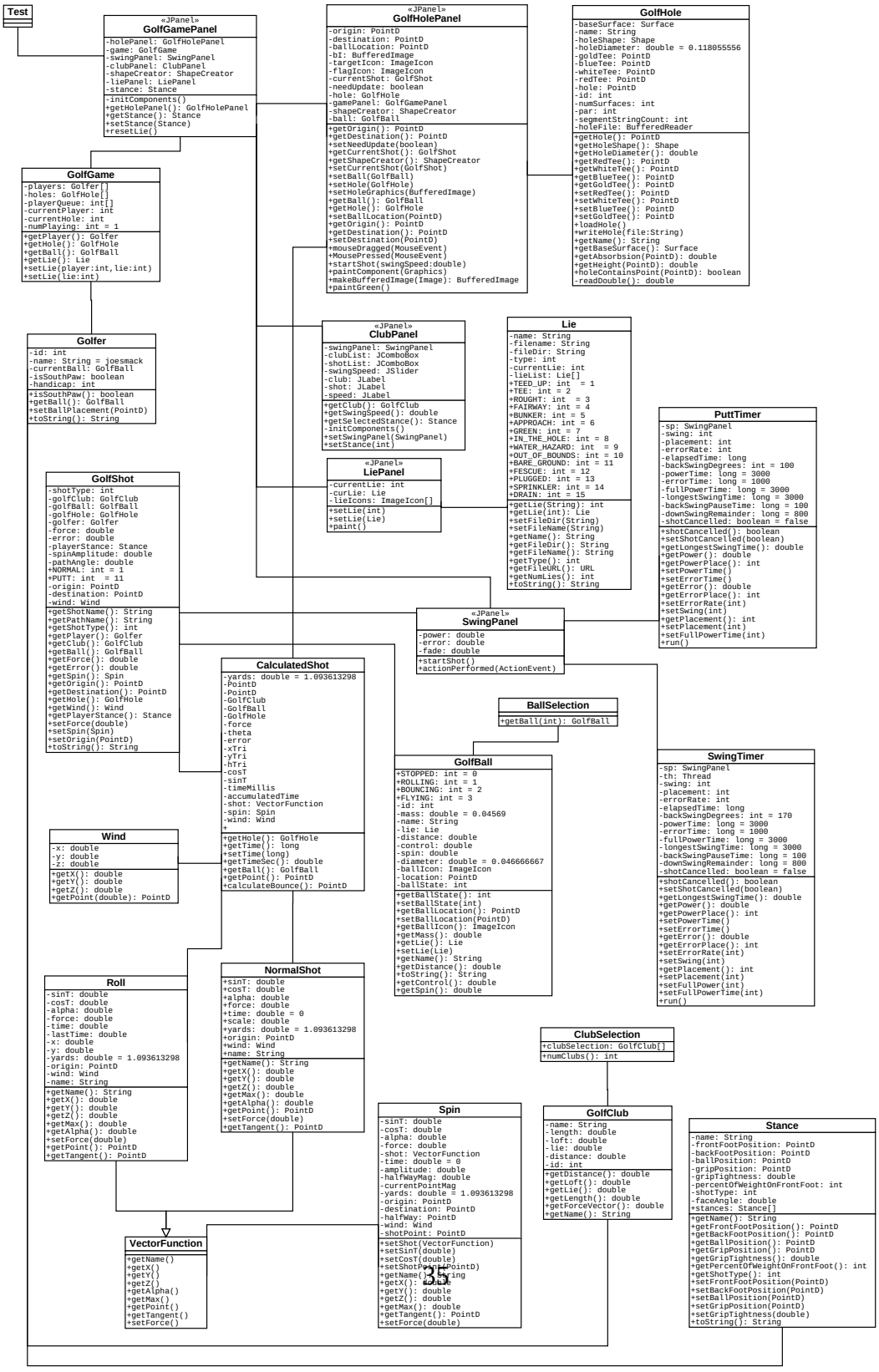

Figure 8: Class Diagram for Golf Shot

3.6.2 Hole Creator Object Interactions

Creating holes are described in appendix A.2. The HoleCreator uses a LineSelector panel to control which tool the user is using. The ShapeCreator handles all the events of creating segments. The add menu in the menubar is used to select which Surface they can edit, or which Surface to add a subsurface to. Figure 9 shows the class diagram for the HoleCreator.

3.6.3 3D Object Interactions

The surface vertices are created in the main class. The array of vertices is then passed to the SurfaceCanvas to be painted. The SurfaceCanvas handles all the rotation events and the SettingPanel handles all the input parameters. Figure 10 shows the changes made to the SurfacePlotter project in order to display golf holes. Figure 11 illustrates a hole projected into 3D using the SurfacePlotter.

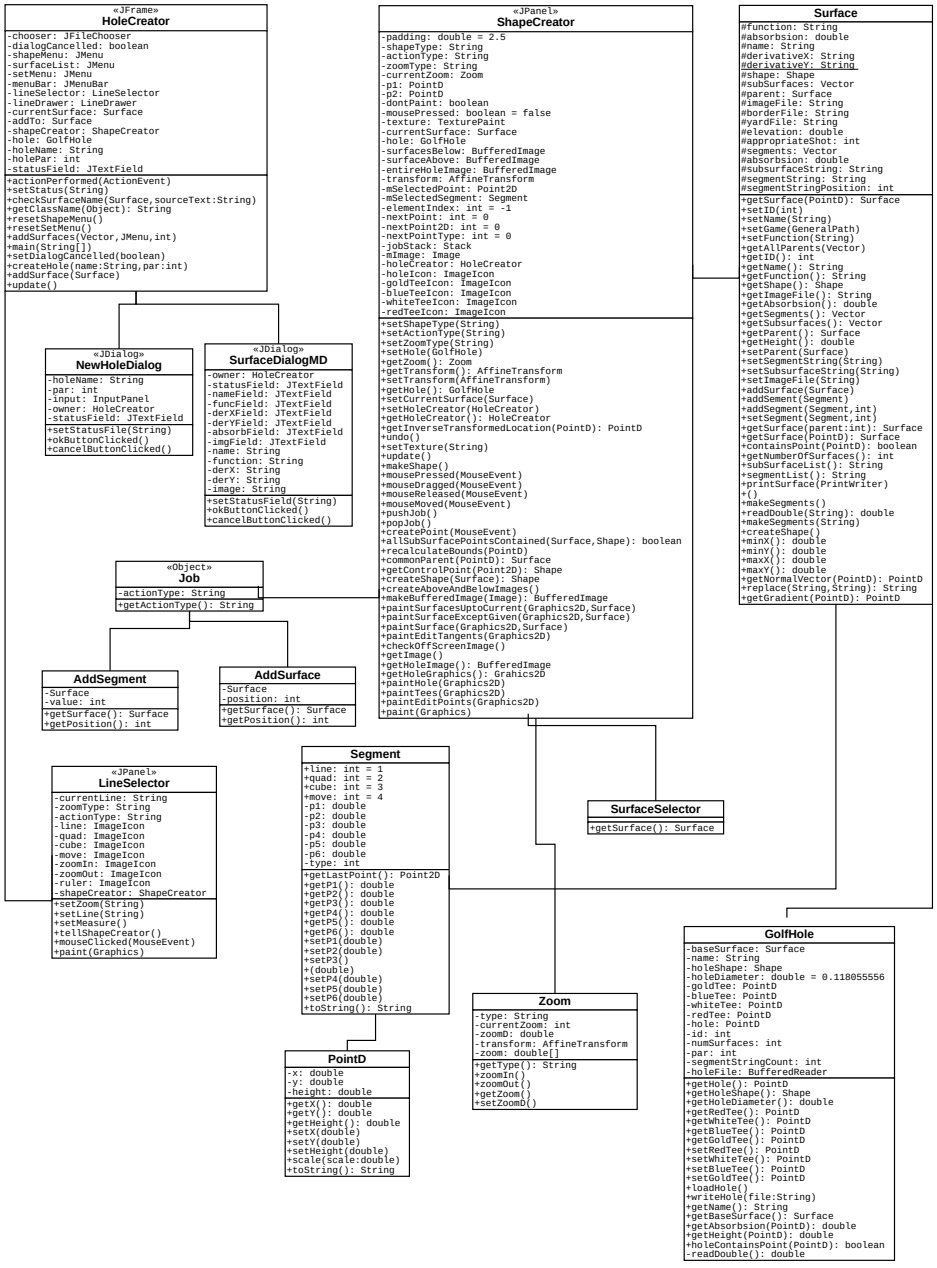

Figure 9: Class Diagram for HoleCreator

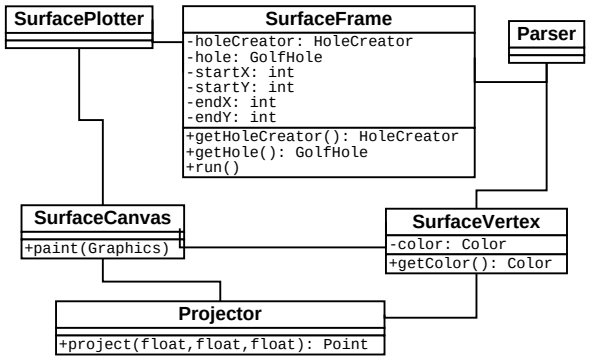

Figure 10: Class Diagram for 3D Engine

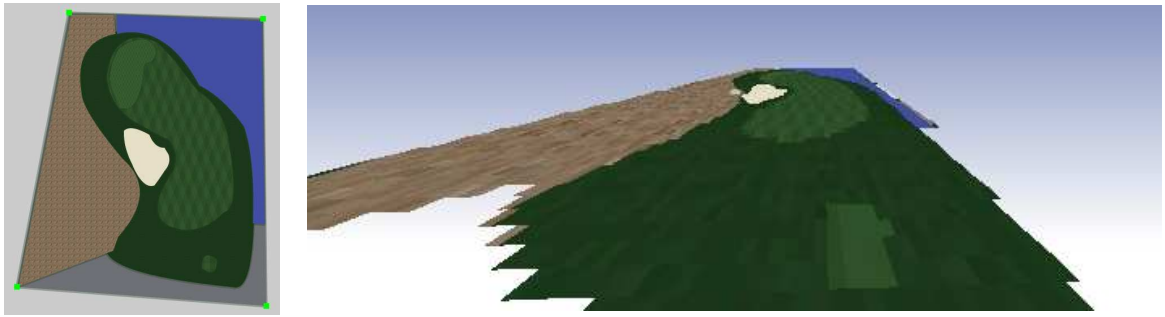

Figure 11: 2D Map Rendered in the 3D Engine

4 Results

The golf shot relies on many parameters including, power, error, spin, wind, and club. The shot may also be altered depending on the type of terrain on which the ball lands. The parameters embedded into the terrain that allow for different shots are absorption and friction. By altering these parameters, the ball will fly, land, and roll differently.

4.1 Flying Golf Ball

For a flying golf ball, all of the above mentioned variables will affect the shot. Since, the shot will become increasingly complex with different variable input, some default values have been chosen for each shot in order to show how the specific variables change the golf shot. Table 1 illustrates the default variables selected for they flying shot tests.

Default Parameters	
Variable	*Value*
Club	Driver
Wind	0 mph
Power	100%
Error	0
Absorption	0.5
Friction	0.1
Backspin	25 revolutions/sec

Table 1: Default Values for Shot Input

4.1.1 Force

Every club has a default force vector. When this force vector is used at 100% the ball will fly further than if hit with less force. The realism that this imposes on a

golf game is that the user may take $3/4$ or $1/2$ swings thus, obtaining less distance on their shot. In golf, it is recommended for high Iron and Wedge play to develop your $3/4$ and $1/2$ swings in order to get precise shots around the green. Table 2 illustrates the distances obtained through each club by altering the force imposed on the ball.

Club	*Distance (yards)*		
	100% Power	*75% Power*	*50% Power*
Driver	282.68	170.26	77.17
3-Wood	271.99	164.29	74.67
5-Wood	259.16	157.04	71.41
3-Iron	233.14	141.99	64.78
5-Iron	219.77	127.59	58.28
7-Iron	176.47	102.58	60.65
9-Iron	132.88	72.32	48.12
P-Wedge	107.20	62.41	40.14
S-Wedge	74.76	43.57	29.36

Table 2: Display of Distance with Variable Power and Club

4.1.2 Backspin

The magnus force imposed on a ball will create greater lift on the ball if backspin is increased. By increasing the backspin on the ball, the ball should go further, and thus decreasing the backspin will force the ball to go a shorter distance. Recall from section 2.2.4 that topspin has not been implemented due to it producing a poor shot. Table 3 illustrates the effects of backspin on the ball.

4.1.3 Absorption

When the ball hits the ground, the impact reduces the force imposed on the ball. For a very hard terrain such as a cart path or rock, the absorption on the ball is not

Club	*Distance*		
	50rpm Backspin	*25rpm Backspin*	*0rpm Backspin*
Driver	304.97	282.68	239.0
3-Wood	295.67	271.99	230.0
5-Wood	283.84	259.16	218.32
3-Iron	258.57	233.14	195.4
5-Iron	244.5	219.77	185.59
7-Iron	198.9	176.47	148.43
9-Iron	151.66	132.88	111.48
P-Wedge	123.23	107.20	89.72
S-Wedge	86.81	74.76	62.44

Table 3: Display of Distance with Variable Spin and Club

as great as it is for a soft fairway. Each Surface is assigned an absorption level. The absorption level can change under different conditions. For example, if the terrain is wet, the absorption is heightened. Since these values can be arbitrarily assigned, the general practice is to assign water and out of bound regions to 0 absorption. That is, they completely absorb the ball upon impact. Surfaces such as bunkers and tall grass are given absorption rates of approximately 0.2. Fairway, greens, and tees are given an absorption of approximately 0.4 and hard surfaces such as rocks are given an absorption rate of 0.6. Table 4 illustrates the effect of a ball landing on a Surface with different absorption levels.

4.1.4 Friction

As the ball rolls along the ground the ball will slow down differently under surfaces with different friction levels. The ball tends to roll further on pavement in comparison to a fairway. Also, the friction of the rough is greater than that of the fairway and will tend to slow the ball down more. The friction level in this game corresponds directly to how much force will be removed from the ball at

	Distance		
Club	*0.2 Absorption*	*0.4 Absorption*	*0.6 Absorption*
Driver	229.62	255.91	336.31
3-Wood	222.57	247.01	322.24
5-Wood	214.21	136.48	305.51
3-Iron	196.96	214.69	271.53
5-Iron	180.41	194.1	240.03
7-Iron	150.75	158.1	187.79
9-Iron	119.02	121.2	150.64
P-Wedge	99.17	98.95	119.65
S-Wedge	72.45	70.34	81.58

Table 4: Display of Distance with Variable Landing Surface Absorption

every interval. A fast Surface will remove 0.1 friction at every interval and a slow Surface will remove 0.5. Table 5 illustrates the effect of the distance of the ball landing on surfaces with different friction levels.

	Distance		
Club	*0.2 Friction*	*0.4 Friction*	*1.0 Friction*
Driver	278.65	276.66	275.45
3-Wood	268.23	266.36	265.27
5-Wood	255.79	254.11	253.1
3-Iron	230.43	229.07	228.29
5-Iron	208.68	203.13	199.8
7-Iron	168.58	164.61	162.24
9-Iron	127.88	125.38	129.93
P-Wedge	103.73	102.03	101.01
S-Wedge	72.95	72.05	71.52

Table 5: Display of Distance with Variable Landing Surface Friction

4.1.5 Wind

The distance of the ball can be heightened when the wind is behind the ball. Wind could also push against the ball, forcing it to lose distance or it could push the ball from the side altering its direction. Table 6 displays the distance of the driver with variable winds. Figure 12 illustrates the resulting position of the ball under these wind conditions. The numbers in the figure refer to the corresponding tests in table 6.

Wind	*Magnitude*	*Direction*	*Distance*
#1	0mph	-	282.31
#2	8.5mph	N	310.44
#3	8.75mph	S	253.92
#4	9.75mph	W	285.07
#5	9.75mph	E	285.57
#6	14.95mph	SE	259.4
#7	17.85mph	NW	327.92

Table 6: Display of Distance with Wind Speed and Direction

4.1.6 Error

In golf, every shot is a potentially good or bad. In order to obtain a good golf shot, the player must hit the ball the way they want to. The input of the SwingPanel for this parameter is taken in as the error. As the club makes contact with the ball, the club could be square or facing left or right. If the ball is hit square to the target, it is considered a good shot. However, when it is not square the ball will tend to go left or right according to the direction of the clubface. Figure 13 illustrates the position of the ball when struck with different error rates. The balls on the left are caused by a -15° error rate. The balls down the centre have no error and the balls on the right are due to a 15° error rate.

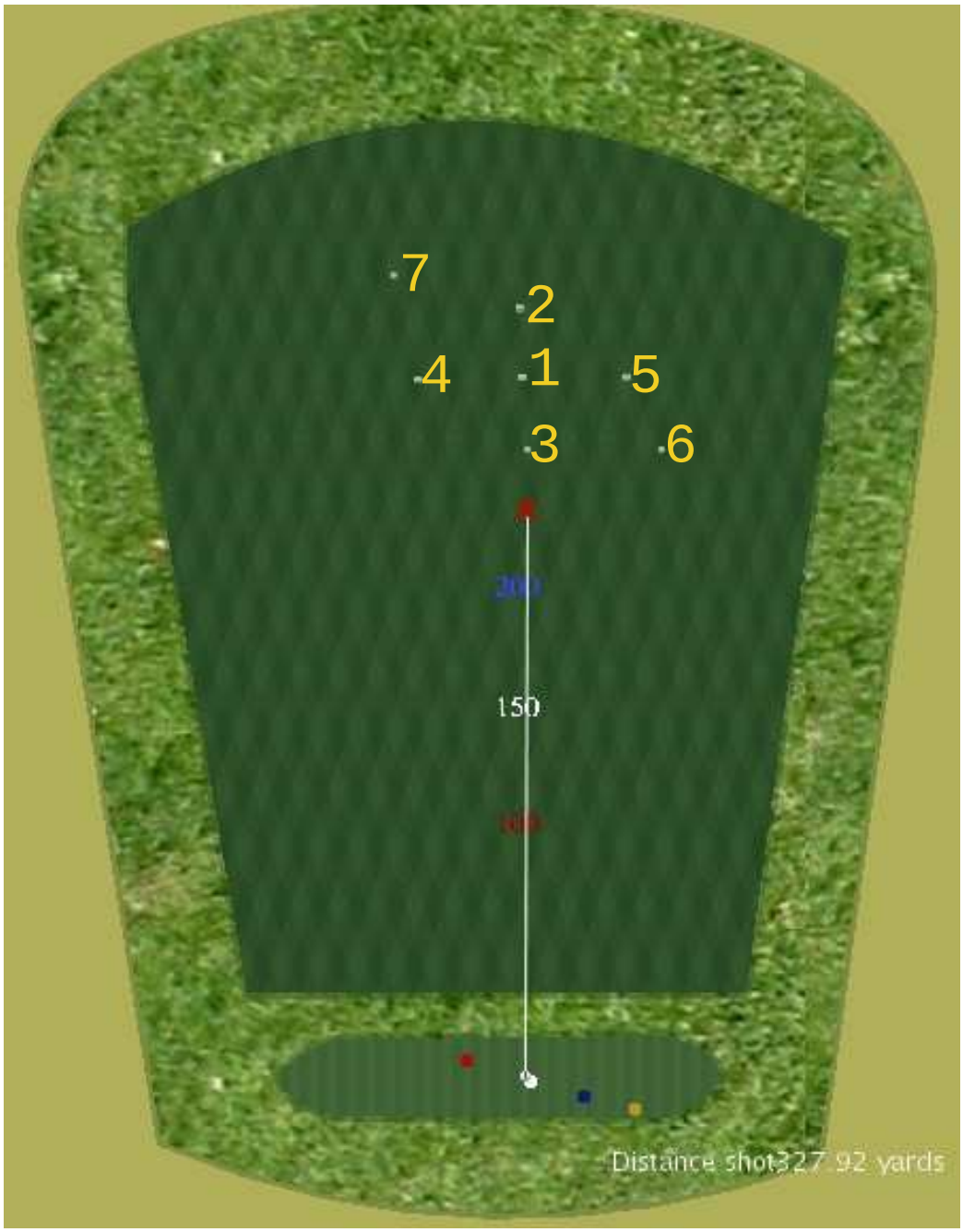

Figure 12: Display of Wind Displacement for each Wind Trial

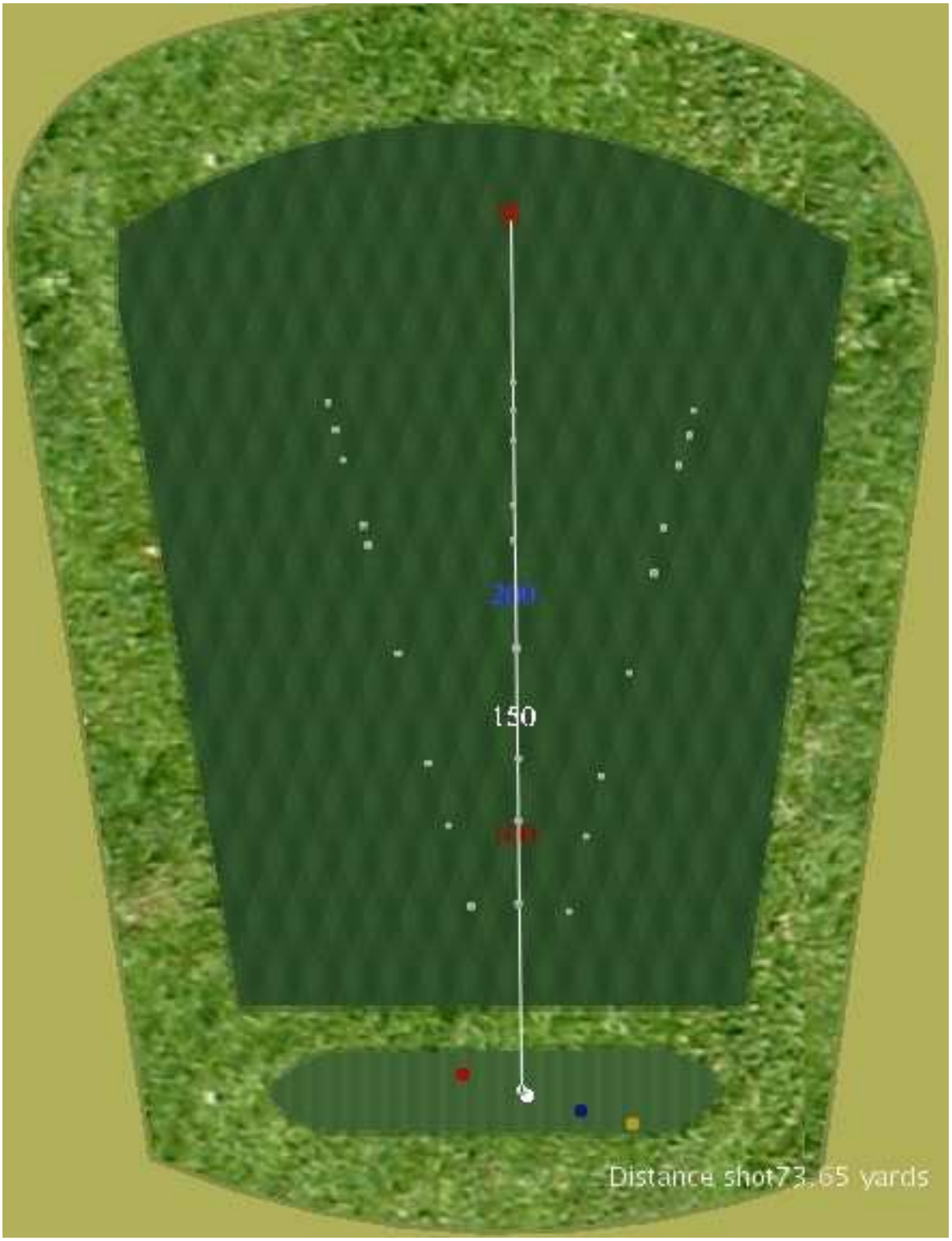

Figure 13: Display of Position with Variable Error and Club

4.1.7 Randomness

The ball bounces at the first point when its height falls below the terrain height. Randomness in the game can be implemented by changing the timer increment from its regular 0.1 second interval to update based on real time. With a 0.1 time interval, the ball will land in the same place because the ball will always fall below the terrain at the exact same time. With a random time interval the ball will fall below the terrain at a different time because the time intervals are no longer uniform. If this time is slightly different when the ball falls below the terrain, the ball will land on a different point and bounce on a different tangent plane. Table 7 shows the drive values with random time intervals.

Trial	*Distance*
#1	283.96
#2	282.59
#3	287.36
#4	283.07
#5	283.15
#6	283.19
#7	284.4
#8	283.92
#9	285.14
#10	284.52

Table 7: Drive Values with Randomness

4.1.8 Fade and Draw

As mentioned before the sidespin on the ball will cause the ball to curve towards the target. Figure 14 illustrates three different positions of the target. Figure 14.a represents a perfect draw or fade which occurs when the target is selected to be

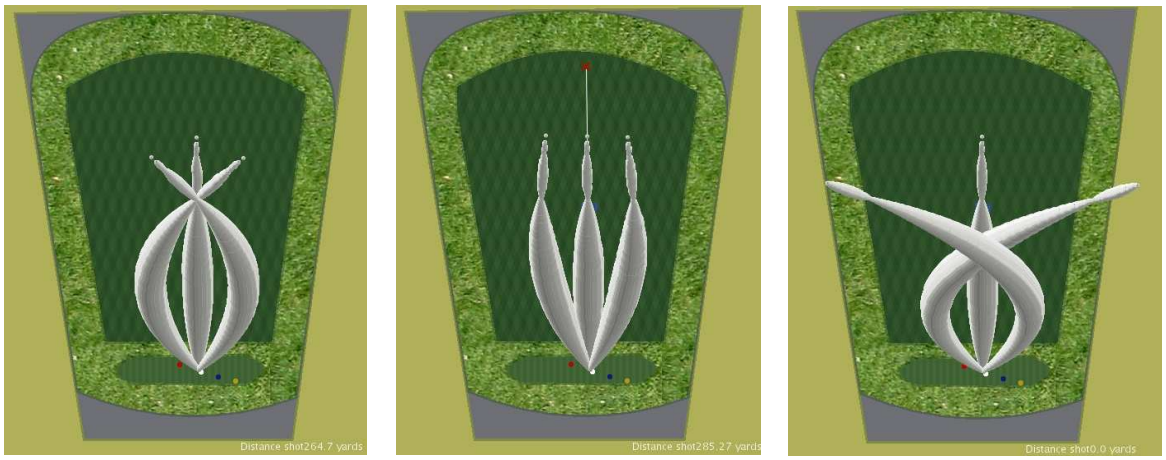

Figure 14: Draw and Fade with Different Target Locations

the point where the ball first lands. This target is ideal because it allows your ball to come in and roll to the side rather than straight. Figure 14.b uses a target that is positioned far beyond where the ball would first land. When this target is chosen, the ball will not be able to completely fade to the target before it bounces. A scenario such as 14.c will occur when the target is selected too close to the origin. The ball will fade to this target too quickly and end up beyond the target before the ball even lands. The result in golf terminology is a slice or hook.

4.2 Rolling Golf Ball

Swing speed is used in putting to control the distance of a putt. In real golf this is called weight. The game implements the fastest swing speed to be 0 and the slowest swing speed to be 100. The faster the swing speed, the further the ball will roll. Table 8 shows the default values for the rolling ball test.

Default Parameters	
Variable	*Value*
Club	Putter
Wind	0 mph
Power	100%
Error	0
Fastest Swing	0
Slowest Swing	100
Distance	Yards

Table 8: Default Values for Putt Input

4.2.1 Friction

As the ball rolls along the ground it will slow down differently upon surfaces with different friction levels. The ball tends to roll further on pavement in comparison to a fairway. Also, the friction of the rough is greater than that of the fairway and will tend to slow the ball down faster . Table 9 illustrates the effect of the distance of a putt when the ball rolls on surfaces with different friction levels and variable swing speeds. Figure 15 shows the difference between distances for a Surface with 0.1 friction(left) and with 0.8 friction(right).

	Distance(yards)			
SwingSpeed	*0.1 Friction*	*0.2 Friction*	*0.4 Friction*	*0.8 Friction*
10	165.06	83.05	42.05	21.51
25	137.56	69.23	35.06	17.94
50	91.74	46.18	23.4	11.99
75	45.92	23.14	11.75	6.04
90	18.42	9.31	4.76	2.47

Table 9: Display of Distance with Variable Surface Friction and Swing Speed

a b

Figure 15: Result of Applying Different Friction When Putting

5 Conclusion

5.1 Summary

It is apparent that through the results, the largest affect on the distance of the ball can be attributed to power. Backspin also plays a major role in the distance of the ball. Since power and spin are input from the user, the game allows the user to have most of the control for distance. Another significant influence on the distance is the terrain absorption. Since the ball will be absorbed more in sand or tall grass it is common to see the ball not make it much further than their original landing positions. The player must be careful when landing on the green with a lot of force as it would be likely to travel right over. Wind causes the ball to be offset in the direction of the wind, generally within 20 yards of the target.

Surface friction does not affect the flight of the ball, but when the ball starts rolling, friction matters a great deal. Friction is the most important force when it comes to putting. A lot of friction can slow the ball down quickly, whereas little friction makes for fast greens. In professional golf tournaments, it is common to hear about how fast the greens are playing. This game can use a friction level of about 0.1 to simulate fast greens. A friction level of 0.2 would be twice as slow, this would make putting much easier to control and good to practice on.

5.1.1 Future Work

The immediate improvements on the game are very minor. For example, in this game simulation wind has a constant displacement in every direction for constant magnitude. In reality, wind has more of an affect if it is against the trajectory of the ball than if the wind is with the direction of the ball. It would also be nice to

have an algorithm that would allow the ball to roll around the hole as it does in reality. The physics need constant tweaking to ensure greater realism. When a new Surface is created such as a rock, the absorption and friction variables must be set precisely in order to preserve the realism. Testing has shown many variables which should be changed to display better bouncing and rolling. Lastly, backspin has not been implemented under rolling. When the ball begins its roll with lots of backspin, it will roll backwards. This process would be tough to animate without a spinning ball as it would look like the ball is being dragged backwards. In the future, the game should have a 3D spinning ball as well as the ability for it to spin backwards.

The game needs to disable appropriate ActionListeners when necessary. This will prevent the occurrence of two Graphic objects being painted at the same time. When this occurs, they often paint over each other.

The most important objective is to get this game online. It is not until the game is online that the popularity of it can spread. For this reason, the game has always been geared towards an applet rather than an application. It is still undecided whether or not sockets or RMI will be used to communicate client and server operations. Sockets are good because they allow complete control over how objects are created and passed between each other. However, small changes might prove to be very tedious to update. I have not yet learned RMI, but I will be taking a course on it in the fall. It is then that I will decide which way to implement the server.

The database should not prove to be difficult to assemble as there has already been successful communication between Java and MySQL. However, the database should be put off until all variables are accounted for, so that tables need not be updated. Golf equipment, as well as shots, and player information will be stored in the database.

Verification systems will be put in place so that the server recognizes any spoofed packets and discards them. Also, systems will be put in place to ensure that the user may not be able to alter their shot input before it is sent to the server. Security on the server is most important as the user should have very limited access to the database. For this reason, only the server will be able to talk to the database by computing the user input. For example, the user will only send across the shot information that is necessary. The server will then send back information such as wind which will be server controlled. This will allow the user to rely on the server to complete their shot. Also, the user cannot account for exact wind while making their shot. If the client sent the server the wind, then they would have full control over the shot input. By talking back and forth, the server can calculate the shot before the client could possibly know where the ball would end up. When the shot is finished the server will check to make sure the client ended up in the same position which it calculated. If not, the server will log this information and handle it accordingly.

The online simulation will contain many aspects of golf which will allow the user to practice. There will be at least one putting green, chipping green, driving range, and practice hole so that the user may be able to get a feeling for the physics of

the game by practicing with their equipment.

The 3D engine can plot holes already, however, it uses quadrilateral shapes that it projects without much control. The next step in the 3D engine is to select the points where it projects more precisely. Instead of choosing colours at points and filling shapes, the 3D engine will select the points over intervals of its texture map. That is, if the texture map for the Surface is 10x20 pixels, the 3D engine will plot points that are every 10 pixels along the x-axis and every 20 pixels along the y-axis. The engine will then transform a copy of the texture map to the transformed 10x20 shape and paint it. This will ensure much better graphics, however, it will not show up close curves in great detail. The goal of the 3D engine is to keep it robust enough to possibly use in other games. For this reason, the 3D engine should be capable of rendering, as though being driven around in a golf cart. If the engine can make it that far, then other games which include cars, or ships could also be created. This combined which the physics of projectiles should allow for easier future game development.

Multiplayer functionality will be added to the game so that users may play with each other in fun, friendly games. As well, tournament functionality will be added so that players can be ranked amongst others. A point system will be used as the game currency to buy clubs, balls, and any other equipment. Points will be awarded to users who enter tournaments according to their rank. It is also possible to award golf clubs or other prizes.

The HoleCreator will be put online so that users can save their holes. Points

may be awarded to people who create holes. There may be some control over who can make holes. For example, the ability to create a hole may be awarded as a tournament prize for low ranking players. By using this prize they can get points. Otherwise, they may settle for a prize of lesser value. The aspect of hole creation is difficult to maintain as people may copy holes that are already done or they might make bad holes. It could prove to be very tedious to check every hole.

Another possible way to earn points is to have courses which people must maintain. If they maintain well they will be given the option of maintaining more courses. However, if they maintain poorly, they could receive a temporary ban from maintenance. Maintenance could be things such as setting the pin and tees for every hole, or possibly some sort of grass cutting, watering, or drainage fixing.

Online forums for game ideas are a definite must. User input on the game will be very highly awarded. When users give good ideas for the game, points will be awarded to them. Bad ideas, or ideas that have already been thought of might still get points, depending on the idea expressed. The best ideas will probably be kept private so that other games do not steal them.

Finally, some form of caddy creation is planned so that players can create (many) sounds for each lie. Some caddies may prove to be mean and taunting, others may encouraging and peaceful. Though it is not yet decided whether or not to allow submission of these sounds or just submission to their URL's.

6 Known Bugs

6.1 Overlapping Surface Bug

Sibling surfaces occur when two surfaces have a common parent. There is not yet overlap detection for sibling surfaces. The reason there is no overlap detection is because if the two surfaces did not overlap, then it would be likely for there to be a gap between them. Consider the case where two sibling surfaces, rough and water, are added to the out of bounds region. If there lies a gap between these surfaces, then the ball could roll out of bounds when it is rolling towards the water from the rough. This would cause a stroke penalty and force the player to begin at the origin rather than the point of entry. Overlapping these surfaces prevents such an occurrence. When the game searches for the Surface which the ball resides on, it looks in the order that the surfaces were created. Painting is also done in that order to ensure subsurfaces are painted on top of their parents. If the rough was created before the water, then the ball will find itself on the rough region. But the hole will paint water over top of the rough making it appear that the ball is in the water region. The only way to fix this, is to minimize the overlap of sibling surfaces without causing gaps. In the future, sibling surfaces might share segments between themselves to allow for perfect overlap.

6.2 Undefined Surface Bug

There is currently no detection or implementation to allow two regions to flow into each other. Consider two parallel planes $Z = 0$ and $Z = 1$. If the first plane is defined over region R_1 and the second is defined over region R_2, then when the ball travels between the regions, if its between 0 and 1, then it would travel

through a void where no terrain exists. However, the ball bounces on the first detection that it has fallen below a surface. If the ball is between 0 and 1, and just entering R_2, then the ball is definitely below $Z = 1$ and will bounce on the first point it calculates. The balls position will be set to the ground level at that point and continue from there. The bug is not in the implementation but rather that there is no terrain defined in some regions.

6.3 Black Surface Bug

This bug is caused by an exception for converting images from RGB to ARGB in the guts of Java. It is a very inconsistent bug and it is hard to duplicate the scenario for which it occurs. Most of the time, image loading works perfectly. It is called the Black Surface Bug, because if the texture map does not load properly, it paints the Surface black.

A Terminology

For a complete list of golf terms and specific golf information such as ball mass and hole diameter, please view
`http://www.leaderboard.com/GLOSSARY.HTM`.[1]

For a complete list of different shots and how to hit them, as well as rules concerning penalties and game play, please consult
`http://www.surreyladiesgolf.co.uk/CRAIGS%20TIPS.pdf`.[2]

B Running The Game and Hole Creator

B.1 Running the Golf Game

To run the golf game type:

appletviewer game.html

B.2 Running the Hole Creator

To run the Hole Creator type:

java HoleCreator

C User Interface

C.1 Golf Game

C.1.1 Select Target

On the GolfHolePanel there is a red X which represents the target. There is a white line which connects the ball to the target. For a straight shot, the target can be placed anywhere in the direction which you would like to hit the ball. If you wish to fade or draw the ball, the ideal target would be where the ball lands. This shot is much more difficult as you would have to estimate how far the ball will go when you hit it as hard as you would like. In general, it is safer to play a straight shot.

C.1.2 Spin

The default spin on the ball is 25 revolutions per second in backspin. The user may choose to increase this to obtain more distance for their shot. They may also choose to decrease the spin to get lesser distance on their shot.

C.1.3 Club

The most important aspect to a golf shot is choosing the right club. Since the objective in golf is to get the ball in the hole, users will soon realize it is impossible to get the ball in if they do not hit it far enough. If the user hits it past the hole there is a possibility that the ball will go in the hole on the way by. However, if the ball is going too fast, it will go by the hole and possibly over the green where there could lie dangerous regions such as bunkers or water.

C.1.4 Swing Power

The next most important aspect to a golf shot is the weight. It is possible to hit a club any distance between 0 and the clubs maximum yardage. To get this weight correct the user must select how hard they wish to hit the club. This value is obtained in the players back swing. If the player takes the club all the way back they will use 100% power. The power for the swing is set as follows:

Click the swing button on the SwingPanel

Wait for the back swing marker to go back as far as you would like to hit it

Press the swing button

C.1.5 Swing Error

When a golfer is in their downswing, they must shift their weight, keep their head still, and do many other things perfectly in order to hit the ball square. The error simulation in the golf swing determines whether or not the club is square when it hits the ball. To obtain no error the user must press the swing button as the club makes contact with the ball. If the player presses the swing button too soon, they

will hit the ball to the right. If they press it too late, they will hit the ball to the left. Error is set as follows:

Set the Power (as described above)

Wait for the follow through to be square to the target

Press the swing button

C.2 Hole Creator

C.2.1 Tools

The four hole tools are:

Zoom In

Zoom Out

Translate Picture

Measure Distance

To zoom in and out, first select the appropriate tool. Click on the panel which contains the hole to zoom in or out. To translate the hole, select the translate tool. Press the mouse somewhere on the panel containing the hole. Drag the mouse in the direction you wish to translate the hole. When the mouse is released, the translation will be calculated. To measure the distance, select the distance tool. Press the mouse on the origin and drag the mouse to the destination. As you drag the mouse around the distance will be output in the console.

C.2.2 Line Tools

The line tools are:

Cubic To

Quad To

Line To

Move To

To use the Cubic To tool, first select it and then click on the hole map where you wish to add the cubic curve. To use the Quad To tool, select it and then click on the hole map where you wish to add the quad curve. To use the Line To tool, select it and click on the hole map where you wish to add a line. These three tools will add their respective segments to the shape of the hole by appending a new line from the last segment's location. That is, when you use one of these tools they will add a new line after the one which was last added. The Move To tool will move the very first point in the Surface to wherever the mouse is pressed and if so, dragged.

C.2.3 New Hole

To create a new hole click File $\rightarrow$ New Hole. The dialog box will ask you to enter the name and par of the hole. Once entered, a default Surface is created which will represent the out of bounds region. The user should then use the Line Tools to create the out of bounds region. Once created the user may add a subsurface to the out of bounds region.

C.2.4 Adding a Subsurface

To add a subsurface click Shape $\rightarrow$ Add To and then the Surface for which you want to add a subsurface. Originally, the only Surface to be added to is the out of bounds region. If you wish to follow a convention, add a rough Surface to the out of bounds region and then add all the major hole components to the rough. For

example, add rough to out of bounds, add fairways, bunkers, and tees to rough. Add an approach to the fairway and add the green to the approach. Regions such as water or fescue may be added to the rough or out of bounds region. When adding a subsurface it is important to remember that you cannot drag points outside of the parent's Surface. For example, if fairway is a subsurface to rough, then all the fairway points must be contained inside the rough shape. The ShapeCreator will check this automatically and prevent the user from such a violation. However, it is possible to create two surfaces which overlap. This is mentioned in the bug section and is the case of sibling surfaces. That is, if water and rough were added to the out of bounds region, they could overlap. This would not be a big deal, however, the point detection on the landing and the surface rendering orders clash so that it is possible to land in what you would think to be the water region but the ball would be detected under the rough region.

C.2.5 Setting Tees, Yard Markers, and the Pin

Each hole allows the user to set the red, white, blue, and gold tee positions. These locations are where the player would tee off from when playing from those decks. The yard markers must be measured with the measuring tool and placed appropriately. The hole must be placed on the green. It is possible to set the pin off the green, however, the game zooms in on the green as the player gets closer to the hole and this functionality would not occur with other surfaces.

C.2.6 Naming Conventions

In order to create proper subsurfaces which will update the LiePanel in the game, the creator of the hole must create surfaces with names that allow the Lie class to detect the type of lie based on the name of the Surface.

red, white, blue, and gold tee decks must begin with ”tee”

all rough regions (blue grass regions) must begin with ”rough”

all fairway regions must begin with ”fairway”

all bunker regions must begin with ”bunker”

all approach regions must begin with ”approach”

all greens must begin with ”green”

every hole (cup /pin) must begin with ”hole”

all water hazards must being with ”water”

all out of bounds regions must begin with ”out of bounds”

all bare ground regions must begin with ”bare ground”

all fescue regions must begin with ”fescue”

all swamp or similar regions must begin with ”bull rush”

If the Surface is any region other than these, or the region fails to comply with those conventions, the default Surface will be rough. This is fine for most regions, however, water hazards and out of bounds regions will not cause stroke penalties if the conventions for those regions are not strictly followed.

D System Requirements

This project was done in Java using J2SDK1.4.1 for Linux. The Java runtime environment can be obtained from `http://java.sun.com`.

Surface Plotter Version 1.30b2 was used to create the 3D engine and can be obtained from `http://www.fedu.uec.ac.jp/~yanto/java/surface/`

This project was tested under Debian Linux stable version, on and Intel Pentium 4, 1.9 GHz with 256MB RAM.

References

[1] Camber Point C/O "Uniform Resource Locators (URL)," http://www.leaderboard.com/GLOSSARY.HTM; accessed July 25, 2003.

[2] Craig Butfoy, "Uniform Resource Locators (URL)," http://www.surreyladiesgolf.co.uk/CRAIGS%20TIPS.pdf; accessed July 25, 2003.

[3] Yanto Suryono "Uniform Resource Locators (URL)," http://www.fedu.uec.ac.jp/~yanto/java/surface/; accessed August 6, 2003.

[4] David M. Bourg, *Physics for Game Developers*, chapter 6, pp. 106–118 O'Reilly & Associates, Inc. Sebastopol, California, First edition, 2002

[5] Jonathan Knudsen, *Java 2D Graphics* O'Reilly & Associates, Inc. Sebastopol, California, First edition, 1999

[6] James Stewart, *Multivariable Calculus*, chapter 14–15, pp. 871–999, Brooks/Cole Publishing Company, Pacific Grove, California, Fourth edition, 1999