Project Codename: Swordplay

Project Members:
desilver
dignatof
lquirk
mkatzour

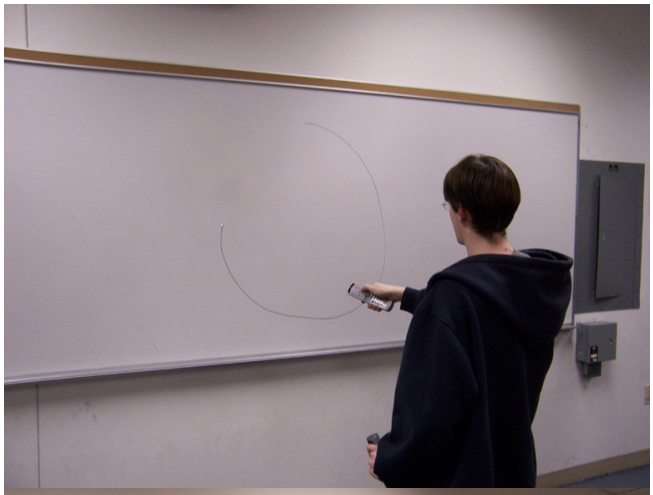
1. Storyboard and Conceptual Images

Lincoln demonstrates the basic concept of the game. A sword and shield appear on the screen, bound to the location and orientation of each of his wanda controllers.

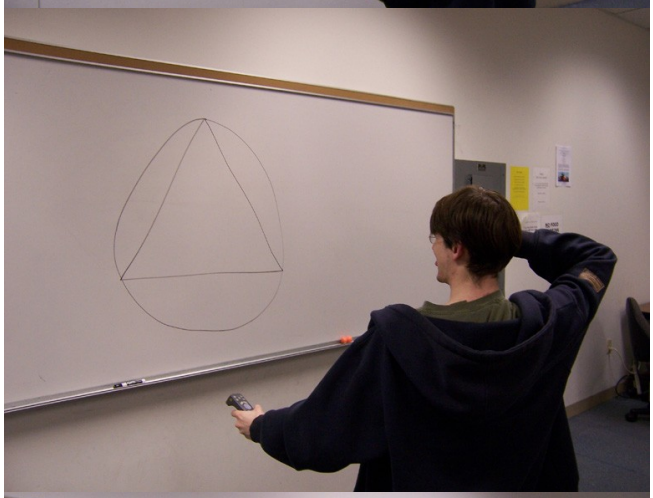
Lincoln swings his sword at a fearsome foe.

A side view of Lincoln in action.

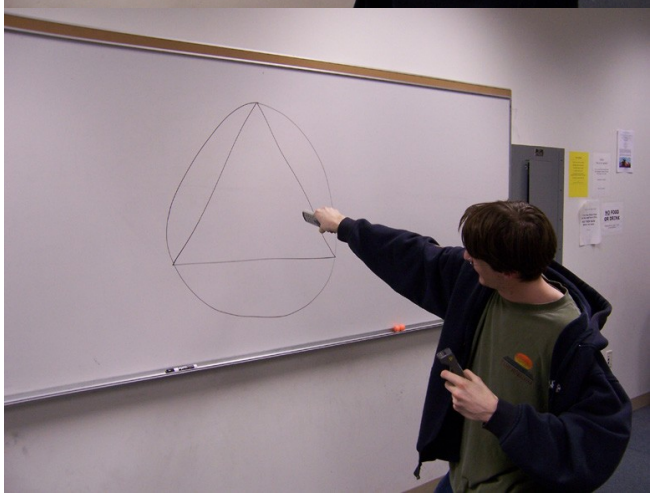
Lincoln begins casting a spell. A button is pressed to initiate the spell, at which point, the tip of the sword draws onto the screen.



This particular spell has the user trace a circle with a triangle inside it. After successfully casting…

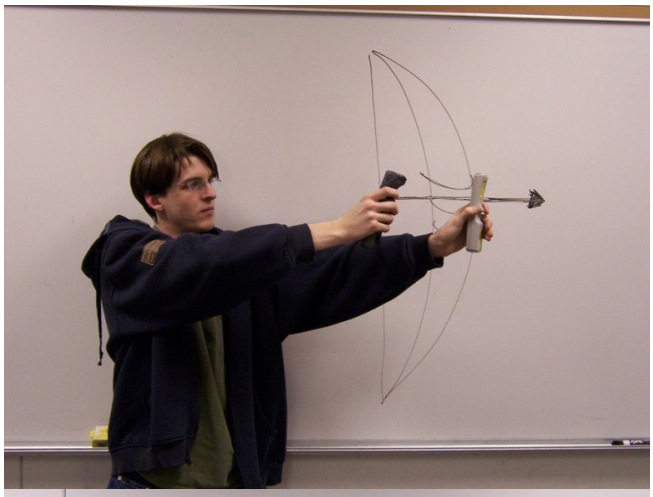


Lincoln thrusts his sword through the center of the spell, triggering it and launching a projectile forward!



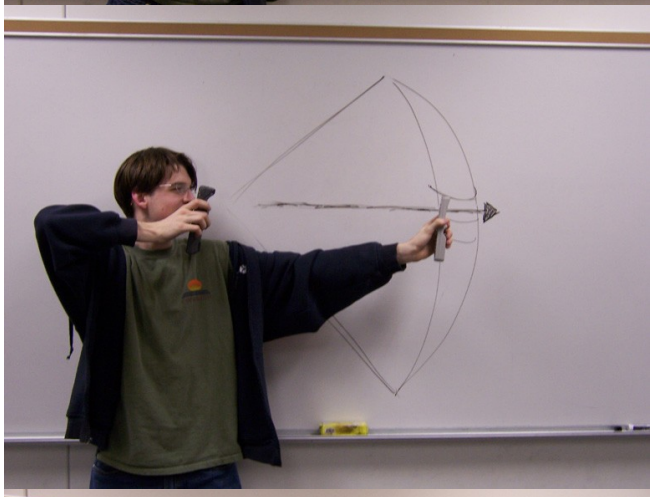
This is an additional feature to be added into the project, the bow and arrow!

Lincoln presses a button to switch modes from sword and shield to bow and arrow. He then brings the arrow to the center of a bow and holds down another button.

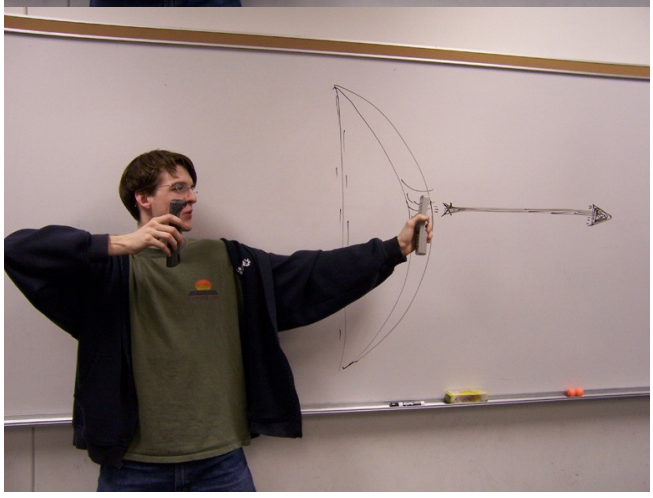


While keeping the button held, Lincoln pulls the arrow back.

When he releases the button…



FIRE!!!



Related works:

Zelda	Zelda has been a major influence gameplay for both the adventure feel and puzzle-solving nature.
Morrowind	Morrowind's combat system takes a first person perspective and involves sword and magical combat, so its gameplay feel resembles ours.
Black & White	Black and White recognizes gestures to cast spells, but is limited in that users can only provide mouse input.
Nintendo Revolution	The Nintendo Revolution controller detects position and orientation, similar to the controllers we will use.
Doom	Swordplay is played in first-person perspective, and thus borrows many gameplay fundamentals from DOOM.
Castlevania: Dawn of Sorrow	Castlevania, for the Nintendo DS, gave us the idea of projecting 3D movements into a 2D space. Its spell system uses several points along a circle, which can be connected with the user's stylus in different orders to cast spells.
Time Crisis	Time Crisis has a pedal, which players use to duck and avoid bullets intuitively. We aim to capture a similar feel of intuitive control with both our "sword" and the ability to duck and dodge attacks.
Star Ocean 3	The Star Ocean series' strength is its battle system. It was one of the first well-known RPG to incorporate a fully real-time battle system. It also features a heavy use of spells/skills that effect areas rather than specific targets or "all enemies".
Surround-Screen Projection-Based Virtual Reality	http://portal.acm.org/citation.cfm?id=166134 – using the CAVE as a virtual reality environment
CavePainting	http://portal.acm.org/citation.cfm?id=364370 – we want to capture the "drawing" feel of casting a spell. This work is in the CAVE and uses CAVE input devices.

2.1 Requirements

Requirements (by priority; 10 is critical, 1 is lowest)

Priority	Description
10	Two controllers/Wandas can be used to directly represent the sword, shield, bow, and arrows used by the player. This includes both the visual aspect of seeing the weapons appear on the screen while being used, and the technical aspect of having them exist within the world/game, capable of interacting with other objects.
10	The controllers can be used for symbolic input. (For example, pressing and releasing a button while using the bow loads and releases an arrow)
10	The controllers can be used to cast spells. This involves tracing out one of many patterns, and having the system appropriately react.
9	Actions taken in the game have visible and/or aural representations. (For example, casting a spell displays some type of animation) Note that flashiness/prettiness is not the point of the game. That factor will only be taken care of if time remains to do do.
9	The player can move through the 3D environment using the controllers/wandas. This includes, at the bare minimum, moving forward/backwards and turning. This can also be extended to include other types of movement, such as strafing.
8	Enemies and other members in the environment exist and can interact with the player, and/or vice versa.
7	Special interaction relating to the sword is implemented, such as a sword hitting another sword or shield. This includes how the UI handles the short loss of control over the sword (from the sword bouncing back)
5	The player can interact with the environment by using spells (For example, by freezing part of the ground to reduce its friction)
4	Enemy monsters have some form of AI (actual or simulated) controlling them during combat.
2	Levels included in the game contain a aspects of puzzle solving and exploration.
1	Plot
1	Spell Backfiring. This is an addition that makes a spell backfire on the player if improperly cast.
1	Additional AI for enemies out of combat. This includes a method of detecting the player's presence.

2.2 Design

Our main requirements are to get movement, combat, and spell casting working. Movement will be done using the remote controls to input symbolic movement, with wanda providing the interface between the remote and the software. Combat, with swords and bow/arrow will be implemented by mapping the position of the wands to the position of the weapon in the game, using simulated physics to model how the weapons interact with other objects. Spells will transform the input from points to an image and analyze it using “GOCR”, an image comparison library that will match images to predefined diagrams.

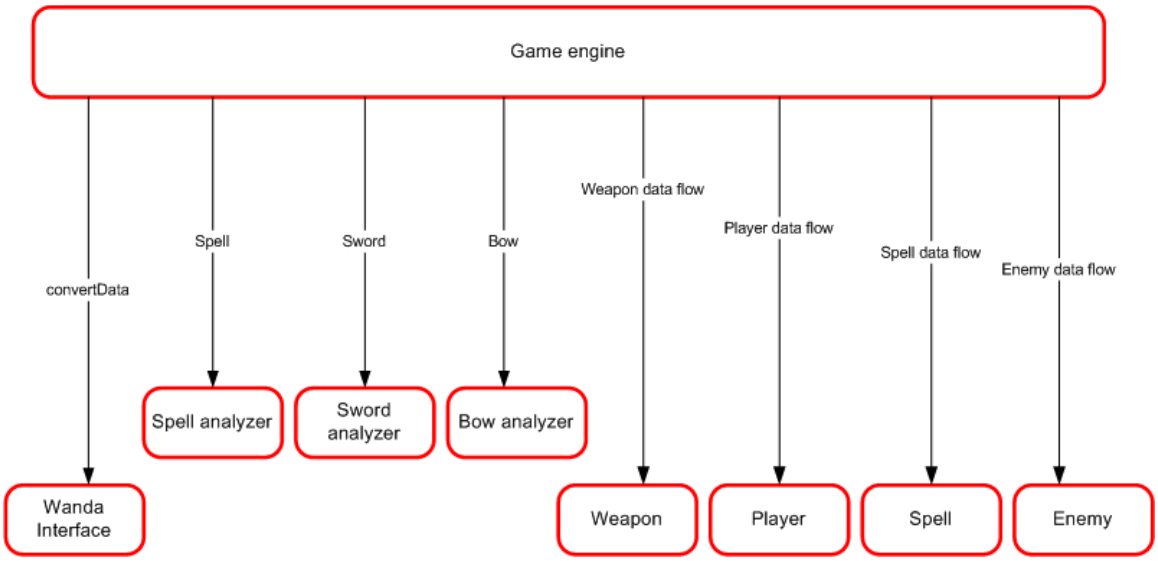
In addition, symbolic actions, such as nocking and releasing an arrow or beginning a spell will be done by a button press. To further explain this concept, the left controller controls a bow, while the right controller is used to nock an arrow and draw the string back. The arrow is nocked upon the button press, and released when the button is released.

Visual representation will be done in the CAVE, but we will use relatively simplistic models or textures until better ones can be found or made. We will be using G3D and VRG3D, which provides us with graphics and collision detection. We have also been offered assistance by the Brown University Center for Computation and Visualization in this process.

Levels and possible puzzles will be done in GTK.

We will be using The CAVE and Wanda remote controls for output/input.

This is a data flow diagram. The arrows are in the direction of function calls, and all interface calls are listed either in the diagram or below. Return values and arguments are described below.



The game engine contains an Input Interface class that, for our project, will be a Wanda Interface. The game will take input in the onUserInput function and pass it directly to the interface in the convertData function. This function will take the data and convert it into either a button press, and return the name of the button. On every simulation step, the game engine will take the input from the two devices and send it to the Wanda Interface once for each device, through convertData. This is a different method with the same name that will convert the data into a six dimensional point.

The wanda interface will return a six dimensional point, or a button press. The game engine will take the point and send it to one of the analyzers, depending on what the last button press was (spell, bow, or none). The spell analyzer will take a point in a sendPoint function, which returns a boolean telling the engine whether the spell is done or not. When that function returns true, the engine will call a getSpell function to get the abstract spell data structure. The sword analyzer will take a point and convert it to a force and torque to apply to the sword object, returning a struct containing this data. The bow analyzer (optional) will take two untransformed points (one for each controller) and return a struct containing the point at the front of the bow, the point where string meets the arrow, and the rotation of the bow around the direction vector. Finally, button presses will either change a flag to {SPELL,SWORD,BOW} or one of several movement buttons.

After getting a completed spell, the engine will create a spell object, sending the type, direction, and the rest of the spell effect is defined inside the spell class. Similarly, after getting the force and torque for the sword, that data is simply passed from the engine to the weapon class which applies the force. The bow data is passed to the weapon and any firing mechanism is done entirely within the weapon class.

The engine will give each enemy a reference to a game state that will allow for any AI that will be implemented. The engine will tell the player how to move.

2.3 Implementation

Our project will be based on the G3D and VRG3D. We will make use of two 6-DOF controllers and the CAVE. The CAVE will be providing us with the controllers. However, there is a desktop version of the cave which hopefully will be fixed before the end of the semester.

A module will compute and provide numeric representations of the controllers' position and orientation. Accessing these values will not consume any significant amount of system memory or processing time.

Performance will not be an issue because the CAVE is run by four separate computers, each with a high-powered graphics card, and our graphics and physics requirements are not heavy.

We are developing on Linux for the CAVE. As such we will be using GDB to debug. The CAVE development machines have a desktop output mode, so that we don't actually have to load the program in the CAVE every time we want to test something.

To implement the spell recognition system, we will be building off of the open source image database management program GOCR. GOCR is intended as an Optical Character Recognition library, but it is sufficiently extensible to be useful in our project. When a user presses the button to start drawing a spell diagram, we will project the tip of the sword onto a plane in front of the user. These projections will be recorded to create an image that will then be fed to GOCR. We will populate GOCR's database with the correct spell diagram images. To compose a three-dimensional spell gesture, we can project to more than one plane (e.g., one in front of the user, one beneath the user, and one to the side of the user).

2.4 Testing

The majority of our testing will be directly related to the UI. Our testing will be done incrementally, relating to how much has been programmed so far.

1) Display the coordinates/orientation of the remotes, and a basic shape in the perceived location of each of the controllers. Make sure that the values update correctly when moved to any direction or location. Try to break the system by flailing the controls rapidly or dropping them. See what happens when a person walks in front of a controller, or what happens when a player spins around. Check if and when the controllers must be re-initialized.

2) Represent the sword and shield as physical objects in the game, replacing the basic shaped used in part 1. First check that the algorithm works in expected situations, such as holding the controller steady, swinging it, thrusting it forwards, and slow to medium speed movements. Then try to break the moving/balancing algorithm using the above methods.

3) For pattern matching with spells, first implement a utility that traces a line in 3D space coming from the tip of the sword object. Set up the interface to display notification of which pattern, if any, is matched. Starting from the simplest of registered patterns, trace out the pattern multiple times and assure that it can be detected. Then trace out the same pattern scaled smaller, then larger, making sure that only reasonable traces are accepted. Try slight variations of the pattern and make sure that the algorithm is neither too strict or loose. Try variations which mostly follow the pattern and at one or 2 points differ drastically, making sure that those are rejected. Try variations which are significantly off, yet resemble the pattern. Finally, try random movements.

The rest of the testing is done using the now tested UI to test individual pieces of the game.

1) Add a single target within swinging range, represented as a box. We will have the onCollision method detect when the sword intersects with the box, and display that on the screen.

2) Add a shield to the box to test its collision. Swing the sword the the shield from various different angles and at different speeds. The goal is to make sure that the sword is properly deflected. Also test striking near the shield without hitting it.

3) Add a sword to the the box to collide with. The sword will move in a preset pattern with a certain amount of force, for example, going back and forth in a 90 degree arc. (We will use multiple different movement patterns.) Make collisions between the player and enemy sword. Also, we will test it by moving our sword very quickly to see if the intersection is still registered, and again, flail the sword around and see if it intersects as it should. Finally, we will interpose our shield and see if that reacts correctly, as well as moving the shield rapidly to break the system.

4) To test movement, we will use the controller to move. To break it, we will try to walk through walls and objects, and make sure that the sword collides with the wall properly. Then we can test movement of the player through the environment to meet enemies. We will test interactions of spells and objects, spells and enemies. Also, we will test the bow and arrow.

5) The lower priority features will be tested by directly observing correctness. Spell-environment interactions will be tested first by applying spells to the targets that should be affected, then applying them to surfaces that do not have a defined interaction. Incremental AI development can be tested by combat with the enemies.

3. Project Timeline

Date	Goals
2/24	Project proposal approved.
3/3	Project proposal revised, experimentation with CAVE, experimentation with GOCR.
3/10	Run our application in CAVE; be able to extract information on the sword's position and orientation, including information on the tip of the sword and its speed.
3/17	Reflect movements of sword onto the screen. Integrate the gesture recognition libraries. Create basic collidable objects in world.
3/24	Be able to reflect intersections of sword and object. Be able to raycast from sword, and recognize gestures such as circles, triangles, etc. “Core” is done.
3/31	Spring break, test as much of core as possible, individual work and research.
4/7	Implement movement, spell visual and game effects, have an enemy that can get hit
4/14	Have an enemy that has preset attacks, bow and arrow
4/21	Write a level utilizing enemies and puzzles
4/28	Everything above completed and thoroughly tested (catch-up week)
5/5	Environment and spell interactions, non-preset enemy attacks, begin plot/motivation development including HUD interaction.
5/12	Finish plot/motivation, polish level, spell backfiring, etc... if there is time.

4. Rubric

Task identification and assignment:

Item	***Point Value***	***Name***
Input/UI	***40***	
Equipped items' position and orientation directly controlled by two wandas	15	lquirk,mkatzour
Sword trace is drawn for the user while casting	5	mkatzour
Sword-tip coordinates used to compose an image for interpretation of spells	5	mkatzour,lquirk
Intuitive transition from sword and shield control to bow and arrow control	3	lquirk,dignatof
Interaction/control scheme for nocking, drawing, and releasing an arrow.	3	desilver
Button presses to user input (e.g. player movement, spell initiation)	4	desilver
Player can dodge attacks by moving or leaning	5	lquirk
Spells	***20***	desilver,lquirk
Display (Spell success/failure is visualized)	2	Lquirk,desilver
Recognition of a spell, given an image	15	dignatof,desilver
Spell interaction with environment/objects	3	Lquirk,desilver
Sword	***15***	mkatzour,dignatof
Display (Sword and shield are displayed)	1	dignatof
Sword/shield collision effects	10	mkatzour
Velocity of sword and shield are tracked	4	dignatof
Bow	***5***	mkatzour,dignatof
Display (Bow and arrow are displayed)	2	desilver
Arrow trajectory and collision effects	3	mkatzour

Item	*Point Value*	*Name*
Enemy	***5***	dignatof,desilver mkatzour
Display (Enemy and its actions are shown)	1	desilver
Attacking (precanned attack sequences)	2	dignatof
Movement (unintelligent)	2	mkatzour
Advanced AI, Blocking player attacks, intelligent path planning	(extra credit)	TBD
Content	***5***	
Level design		mkatzour,dignatof
Textures and animation		all
Game and environment objects		dignatof,mkatzour
Sound Effects (collisions, cast)		desilver
Music		desilver
GameApp / Core	***Required***	lquirk,desilver
Player		
Camera		
Setup & startup		
Game Menu	***Optional***	TBD
Start game		
Calibrate controller		
Administrative	***10***	
Version control repository		desilver
Weekly presentations		mkatzour
Documentation		dignatof
Web		lquirk
Wiki		lquirk

5. Selection of a mentor

Graham has already acquiesced to being our mentor.