T. MARTIN SENNETT
Spring 2011

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Information Sciences and Technology
with honors in Interdisciplinary Digital Studio

Reviewed and approved* by the following:

Carlos Rosas
Associate Professor of Art
Thesis Supervisor

Andrea Tapia
Assistant Professor of IST
Honors Adviser

* Signatures are on file in the Schreyer Honors College.
ABSTRACT

For my thesis, I collaborated with another student (Ryan Stubbe, a non-scholar majoring in IDS) to create a computer game. We wanted to explore several concepts, including mouse-based analog controls, procedurally generated content, and player-driven item creation. Our final product differs greatly from our original vision, but ultimately touches upon all of these ideas in a novel way.

Our work provides an excellent example of two principles key to game design: iteration and synthesis. Over the course of development, we built and discarded many different prototypes, each time discovering new things we liked and disliked. Additionally, we incorporated ideas from other games into our design, combining them in new ways. Following these two principles enabled us to create a fun game that feels both unique and familiar.
TABLE OF CONTENTS

LIST OF FIGURES.............................................................................................................iii

Chapter 1: Project Origins..............................................................................................1

Chapter 2: Development Process.......................................................................................7

Chapter 3: Results and Future Work................................................................................13

Bibliography.....................................................................................................................15
LIST OF FIGURES

Figure 1: Sexy Hiking by Jazzuo .................................................................1
Figure 2: The Jabberwocky prototype control system ............................3
Figure 3: The CRUSH! prototype control system .................................4
Figure 4: Hamurai Snackdown .................................................................5
Figure 5: Captain Forever by Farbs .........................................................8
Figure 6: Gesture detection system .........................................................9
Figure 7: Katamari Damacy by Namco ....................................................11
Figure 8: The final game .......................................................................1
Chapter 1: Project Origins

The oldest inspiration I can remember for this project is a strange little game called *Sexy Hiking*. *Sexy Hiking* was released for PC in 2003 by a mysterious developer known only to the Internet as “Jazzuo”, whose games are renowned for their low production values, sense of humor, and innovative gameplay. *Sexy Hiking* is no exception, presenting the player with crude graphics, strange sound effects, and novel (though buggy) controls.

Figure 1: *Sexy Hiking* by Jazzuo. Source: db.tigsource.com
In *Sexy Hiking*, your character can move left and right and use a pickaxe to scale ledges. This pickaxe is mouse-controlled - it hovers around the character’s body, oriented toward the mouse cursor. To climb, you have to swing your pickaxe around to the top of a ledge and then pull down to hoist yourself up. It is a wholly unique, frustrating, and hilarious experience.

I decided I wanted to make a mouse-based 2D game like *Sexy Hiking* at some point, but a bit more accessible. I really liked the freedom of motion provided by *Sexy Hiking*’s setup, and that the interactions in the game world arose naturally out of it.

I built my first prototype in Game Maker, which is great for getting a game up and running quickly and easily, and which I happened to be very familiar with at the time. Instead of a pickaxe, I gave the player character a sword, with the notion of adapting Lewis Carroll’s *Jabberwocky*. 
In the Jabberwocky prototype, your sword would hover toward the mouse cursor, and when holding down the left mouse button, rotate to aim at it. This translated into the ability to execute some nifty poking and slashing motions, if you knew what you were doing. However, when I put first-time players at the helm, they flailed about helplessly. The click-and-drag system of control didn’t make much sense to people who hadn’t programmed it.

I discarded the Jabberywocky build and began another project in Game Maker that was to be a sequel to a previous game of mine called CRUSH!. In this prototype, you controlled a giant robot whose hands you could swing around and pick up buildings with using the mouse. It was fun to have differently shaped and sized objects to wield, but because you could only hold one
object at a time and they were quickly destroyed, it devolved into a matter of picking up and
tossing everything in sight, and because of the sheer scale of the player character and
environment, it lost all of the precision and nuance I had been going for, and so I abandoned it.

Figure 3: The CRUSH! prototype control system.

After I learned how to work with Flash, I decided to take another crack at the sword idea.
This time I was going to make a sequel to another previous game of mine, *Samurai Smackdown*.
It would feature the titular samurai giving up his violent ways to roam the countryside with his
son, making giant sandwiches to pacify monsters. The name? *Hamurai Snackdown*, of course.
The controls in the Hamurai prototype were a reined-in version of those in the Jabberwocky prototype, with some significant differences. Your sword hovered much closer to your body, and would automatically point at the cursor, allowing you to slash by simply waving the mouse. Holding the left mouse button would lock the angle of your sword, allowing you to execute a poking motion. You could use these skills to chop flying hunks of deli meat, cheese, and bread, and defend yourself from attacking ninjas. Unlike the previous prototypes, which featured scrolling levels, this game was a sequence of single screens.

Players were quite a bit more effective with the Hamurai setup as compared to the Jabberwocky one, but often they would get into a tough situation and start flailing and the sword would not operate the way they expected it to. The controls left more room for error than the Jabberwocky ones, but they still required a degree of precision that was not intuitive nor
especially fun. I made a fairly feature-complete demo of this game, but it met a lukewarm reception from playtesters, and I decided to shelve it.
Chapter 2: Development Process

I had built, tweaked, and destroyed several times over, and still not found the fun I thought I could in the concept of whacking things with a mouse-controlled object. I gave up on it for a while, until the time came to settle on a thesis project.

Ryan Stubbe and I decided to collaborate on a game, and after looking at my previous prototypes and discussing our ideas, we agreed we would try a mouse-controlled platformer beat-em-up.

We liked the ideas of destructible/usable environments and weapon variety from the CRUSH! prototype, and tried to think of some way to work them into the game’s design. I had recently played a very cool game called Captain Forever by Jarrad Woods (aka Farbs), in which you dynamically build and rebuild a spaceship by sticking together modular parts.
I suggested to Ryan that we could apply the same concept to our player character’s weapon, such that its size and shape and handling would always be changing, and players could construct weapons of their own design. We ran with the idea and decided the player character would be an inventor, with an all-purpose mechanical staff you could slot pieces into.

At this point I jumped into coding yet another attempt at mouse-based weapon play. Based on my observations of previous prototypes and brainstorming with Ryan, I decided to take a step away from analog, 1:1 mouse controls.

The new aim was to create more gesture-oriented controls, such that two players making different slashing motions with their cursors would both get the same result, so long as they were aiming in the same general direction.
In any of the prototypes, when given no instruction, players were inclined to click and hold the mouse button when beginning a “slash”, move the cursor in a line through their target, then release the mouse button. I decided to use this setup as a base, so that the system would support players’ natural motions rather than fight them. I coded the new system to detect the point onscreen where the mouse button was pressed, and then to detect the point where it was released, and then to analyze the resulting line between the two to determine what the player was trying to do.

Figure 6: Gesture detection system.
After extensive tweaking and testing, I decided that the main thing this setup afforded was consistency of control. With the previous prototypes, a slashing gesture would need to be different depending on where your character was onscreen in relation to the mouse; to attack, you would have to make tight strokes above and to the sides of your character. With the new setup, the focus became striking through enemies with your cursor, regardless of where your character was. People found this much more natural.

Initially, I built this system with canned actions and animations to simplify things. After establishing what we liked about it, I converted it back to an analog system, so that your weapon once again tracked the mouse, though this time with momentum and smoothed movement. The simplified gesture setup had shown us what we wanted functionally, and so I stuck to that while rebuilding the freedom of motion from previous prototypes.

What we ended up with was a control system based around broad mouse strokes, far removed from the precision required in *Sexy Hiking*. While our system is not nearly as fine-grained (or frustrating), it still provides an enhanced level of freedom over simply having an “attack” button.

Once the controls were established, the rest of the game’s design followed. To facilitate broad strokes, we wanted to maximize the screen area players could move their cursor through, so we positioned the player character on the far left side of the game view. From there, it naturally became a game about going to the right. Enemies would knock you back to the left, and you would smack them to the right. This created a very intuitive metric of progress and a clear, though forgiving, punishment for messing up. It also made the decision to go with procedurally generated levels, something we had been considering, much easier. This meant we would write an algorithm that would create levels for us, rather than design them manually. Procedural generation is much more feasible when creating levels that are basically straight lines, as ours would be.
Our idea for custom weapon construction also shifted away from precision. While we had originally envisioned a mechanical staff with a discrete number of slots for various devices, such an exact system did not fit with the game’s new, looser controls. Instead, we set it up so that things you can attach to your staff just glob on when they make contact with it. This method of constructing a big mass of stuff is somewhat reminiscent of the PlayStation 2 game *Katamari Damacy*.

![Figure 7: Katamari Damacy by Namco. Source: wikipedia.org](image)

In our game, you can only attach things to your staff if you’re holding down the left mouse button. When you release the mouse button everything falls off - you can use this to fling stuff at enemies or to rearrange the pieces. This setup keeps things dynamic and simple, and provides a nice parallel between the character’s action onscreen (holding a spell while swinging the staff around) and the player’s physical action (holding the mouse button while moving the mouse around).

With these new elements in place, we began to rethink our inventor concept. The weapon construction now felt more like magic than science, and the handling felt more like an amateur's than a professional's.
So we came up with a new character - a young apprentice to a kind old wizard. When the game begins, you learn your first spell, which allows you to attach things to your staff. Before you can learn anymore, an evil wizard kidnaps the good wizard. This simple story sets up all of the game’s elements - it gives an explanation as to how you can pick things up with your staff but can’t do anything else, as well as why you need to get to the right. While we had struggled to come up with a story for our inventor when we were still figuring out the control mechanics, the story of our young apprentice came about quickly and naturally because we knew already what the gameplay looked like.

Indeed, once the basic handling and construction of the character’s weapon was iterated to a point we were happy with, the rest of the game fell into place pretty easily. We made enemies that require you to build and attack with a little strategy, and enemies that test your reflexes. We filled the levels with stuff for you to fling around. We made a level generator that puts everything in the right order. Once we had something fun, the rest was easy.
Chapter 3: Results and Future Work

The game we’ve ended up with is a lot different, and a lot better, than my Jabberwocky prototype from so long ago, and even from the ideas Ryan and I had when we began this collaboration. We took inspiration from other games, tried something new, and kept pounding away at it until we got something we liked.

The game can be played right here:

http://personal.psu.edu/tms5162/thesis/play.html

Figure 8: The final game.

While our game is short, usually taking under 15 minutes to complete, it has no filler content and we’re happy with how it’s turned out. The main aspect that could be improved at this point is technical performance – because the game was developed for Flash, and because it uses
so many high-resolution images which are constantly being rotated and transformed, the framerate can dip significantly. But in terms of design, the game explores the concepts we wanted to fairly deeply, in a short span.

That said, the idea of mouse-based magic could certainly be developed and expanded upon. Our story conveniently ends with the certainty that our main character will be learning more spells - perhaps our players would like to as well.
Bibliography


Dave Shaw <http://www.jabberwocky.com/carroll/jabber/jabberwocky.html>


<http://www.jazzuo.com/games/sexyhands/sexy_hiking/sexyhiking.htm>
Vita of T. Martin Sennett
istomsennett@gmail.com ~ http://wherecouldtom.be

Education

The Pennsylvania State University, May 2011
B.S. in Information Sciences and Technology, College of IST
Honors in Interdisciplinary Digital Studio, School of Visual Arts

Conference Presentations


Research Experience

Research Assistant – Coordinate Quest
The Pennsylvania State University, August 2010 – present
Designed and developed a Flash game to illustrate how people can mobilize and
collaborate over social networks and other communication channels. Programmed the
game using ActionScript 3 and created artwork using Adobe Photoshop.

Research Assistant – Case-based and Collaborative Learning
The Pennsylvania State University, May 2008 – March 2010
Developed web-based tools for use in IST classes and research studies. Worked primarily
with HTML/CSS/Javascript, with another programmer doing backend.

Research Assistant – Social Movements and Protests in Virtual Worlds
The Pennsylvania State University, January 2008 – May 2009
Investigated details of specific virtual world protests. Tracked down and interviewed
individuals involved. Wrote case studies relating the events.

Independent Game Development

Deepak Fights Robots
http://deepaksave.us, January 2010 – April 2011
Independently developed Flash game, pending release. Responsible for all art, coding,
and design work.

RunMan: Race Around the World
Collaborated over the Internet with a student developer from Canada to create a PC
game. Responsible for all art, as well as about half of programming and design work.

Other Games
http://wherecouldtom.be, 2005 – present
Created 11 smaller games, including all artwork and code contained therein.