Versus Fighting Game for Nintendo DS: Champion

Ian Roskam
CS 470
April 24, 2009

Table of Contents

Abstract

A versus fighting game pits two characters in a match against each other. The goal of the game is to defeat the opponent by hitting them. The player is able to select the character that they will be playing and in some implementations will also choose their opponent and background. Versus fighters gained popularity during the mid-90s with the release of "Street Fighter 2" on the Super Nintendo Entertainment System. The Nintendo DS is fully capable of reproducing games exceeding SNES quality. Despite the device's capability there is a dearth of fighting games for the system. This project attempts to fill the gaping hole of the versus fighter genre on the Nintendo DS system.

1. Introduction

This game is a personal project that was conceived as a small component of a much larger game project. The only games that I have designed before were as academic exercises. I would like to get into game design and game programming. This project was an introduction to Nintendo DS programming and much more involved than any game I had programmed before.

2. Project Overview

2.1: *Purpose*

The goal of this project is to design and develop a basic fighting game for the Nintendo DS to be reused in a larger game. I enjoy fighting games and I have found that there is a lack of good fighting games for the Nintendo DS. I will also be using this project to learn how to program the DS so I can develop games for the system in the future.

2.2: *Game Overview*

Champion will be a single player game. The player can choose their character and opponent as well as the map they will fight on. There will be eight characters to choose from and seven backgrounds. The graphics component is far too time consuming for me to take on. I lack the artistic skill to draw the required images. To save time all of the graphics will be place holders from commercial games.

When starting Champion the user will be welcomed by the title screen and options to continue. On the character select screen the player will choose their character, opponent, and fight background scene. During a fight the player controls their character using buttons on the DS for input. The opponent character will be controlled by an AI to make the game more interesting.

2.3: *Library*

To program the Nintendo DS I will be using an unofficial library called 'libnds' that is part of devkitPro, an open source multi-platform SDK. The libnds library provides access to the majority of DS functions but is not nearly as feature rich as the commercial NDS SDK. All of the functionality provided by libnds has been reverse engineered from either Game Boy Advanced or the Nintendo DS. As a result some features do not work as well as they should.

The official SDK would be desirable for getting into DS game programming. Unfortunately, Nintendo is rather picky about who gets to write games for their systems. The Nintendo DS developers kit and SDK cost between $2,000 and $10,000 depending on the options chosen for the developers kit.

3. Requirements

The requirements that I set out to accomplish were weak on details. I found that I spent more time during the design phase thinking of optional features. This led to a lack of detail for the implementation of required features. As a result I found myself designing feature implementations along the way that greatly slowed progress. The requirements below are the original requirements.

3.1: *Game Engine*

Required:

- Collision detection is essential to the function of most games and is vital to the versus fighting genre. A double bit mask of the character sprites would allow pixel perfect collision detection but might prove to be too costly on the DS hardware. As an alternative bounding box collision detection could be used but is likely to result in false collisions. If a double bit mask proves to be too costly I may confine the detection area inside a bounding box to limit the number of pixels that are checked and reduce the cost. I will have to play around with the collision detection to find the most effective method but the idea is to have pixel perfect collision detection.
- Basic moves are high punch, low punch, high kick, low kick, duck and block.
- AI is necessary to make fighting the CPU more enjoyable than beating on a mannequin. For a simple AI a finite state machine with a bit of randomness added in should work.
- Loading the selected level and characters to start a match should not be very difficult to implement.

Optional:

- More advanced moves could include grapple, throw, projectile, and weapon.
- Move sequences or combos are a nice feature in any fighting game and add to the value of the game. Since combos are desired but not necessary I will leave them as a first priority option. There are a few ways that I could implement combos. For simple combos a finite state machine made using 'if' or 'switch' statements would be acceptable. The desired route would be to include complex combos with long button sequences and hold timers. For an ideal implementation an event queue would be required.
- Music and sound effects would be a nice addition to the game engine but they are not necessary for the engine to function. If I have time I will add sound to the engine.
- Wifi play is the lowest priority option. The DS has the ability to connect to another DS over wifi and share a small game that can be played by two humans. It would be nice to include multi player wifi but this feature will only be included if all other options are completed first.

3.2: *Characters*

Required:

- A minimum of eight characters will be included.

- Starting with only one playable character additional characters can be unlocked by defeating them in battle.
- A sprite strip that includes every position that a character can be in.
- For the initial build I will be using sprite strips from commercial games.

Optional:

- An additional eight characters could be included for a total of 16 playable characters.
- If items are included character sprite strips will need to include characters with weapons.
- Custom characters with sprite strips.

3.3: *Menus*

Required:

- Title screen menu where the player can select the game mode and options.
- Character select screen where the player can choose the character, opponent and level they wish to play.

Optional:

- In battle menu that can be used to look at the move list or quit.
- Move list where selecting a move will display how the character looks when executing the move.

3.4: *Levels*

Required:

- One level is needed to play matches on.
- A single screen background image.

Optional:

- Additional levels themed for each character in the game.
- Items in the character plane that can be destroyed on impact.

3.5: *Additional Features*

All of the following ideas are entirely optional given that they would probably add unique game play features but they are not vital.

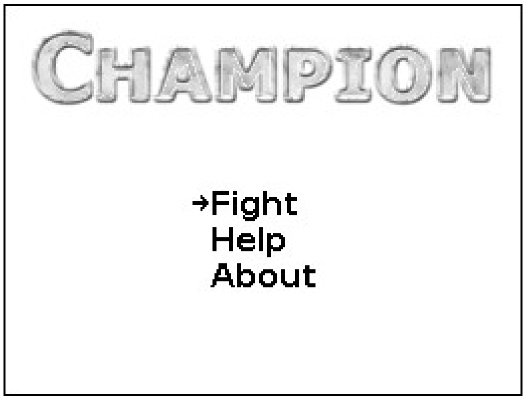
- Equipable items including weapons that have passive stat effects.
- Usable items that temporarily effect the user or opponent. Usable items could include health or mana potions. Using an item would be the same as doing an attack combo.
- Mana based spells are not generally in the fighter genre. Characters generally are able to do some sort of projectile attacks that could be mana based. Limiting the character's mana pool for used to attack or heal adds a bit of strategy to matches.
- Healing spells are something I have understandably never seen in a fighting game. With a high mana cost regardless of success I think healing spells could be implemented in a way that is difficult to abuse.
- Acquiring could be too easy if they are just won from matches so an item shop would be necessary.

If a shop is implemented then reward money would also need to be implemented and item values balanced to reflect the benefits of using an item.

4. Game Design

When Champion loads the player will be greeted with a nice clean title screen. There are three selections that can be made. The 'A' button is used to make selections and 'B' is used to back out. Choosing "Fight" from the menu loads the character select screen. During character selection the player chooses their character, their opponent's character, and a background. After a fight begins the player battles the opponent using attacks and blocking for defense. The first character to have their health depleted, loses. Each fight consists of rounds. The first character to reach two round wins is the fight victor.

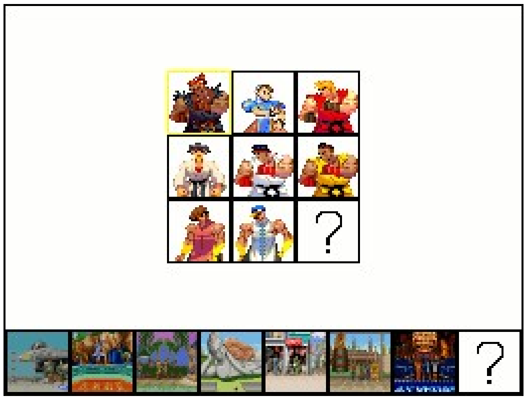
4.1: *User Interface*



The title screen is mostly a background image. The only element of the screen that is not part of the background is the arrow cursor. The cursor is a sprite. Sprites on the DS get their own video RAM space making their manipulation separate from other elements in VRAM.

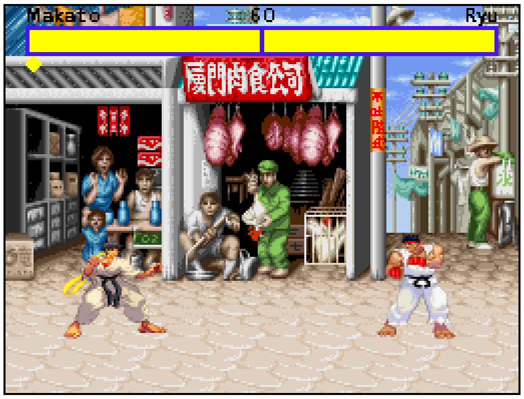
Both 'Help' and 'About' screens are backgrounds displaying the valid information.

Selecting 'Fight' from the menu will bring the player to the character select screen.



The character select screen is similar to the title screen in that only the cursor is not part of the background. The first selection will become player one's character. The second selection is the opponent's character. The third selection sets the background the fight will take place on.

One more press of the confirmation button is required to start the fight engine.



The fight screen displays names, health, and wins of both characters. There is a timer above the health bars. When the timer reaches zero a round is over.

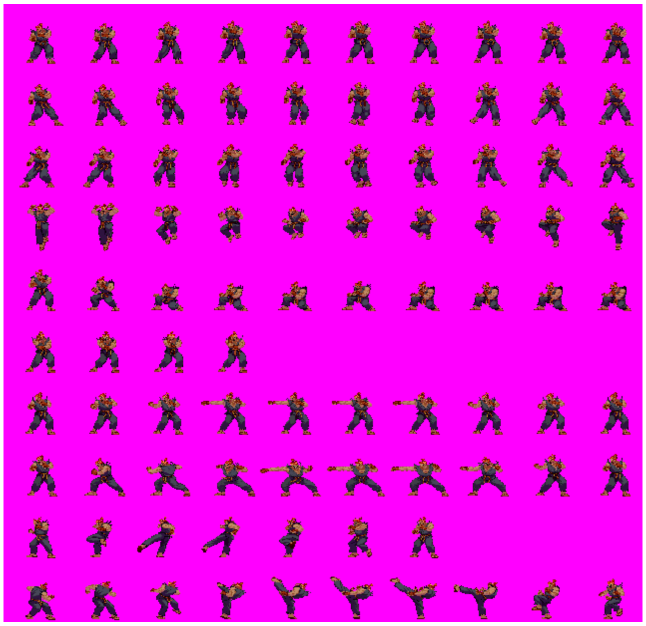
When a character is hit their yellow health bar decreases. When a player wins they receive a win token that is displayed below the health bars.

4.2: *Control Structure*

Transitions between game states is handled by the main class. Taking the limited memory of the DS into account I decided to forgo a menu dedicated menu data structure. Each state has its own class with a start function that begins a loop and returns the next state when finished. The state control is a switch of the current state. When a class' loop returns the object is destroyed. The next call of the control begins the next state.

4.3: *Sprites*

To animate each character ninety animation frames are required. Each animation is composed of at most ten frames. Frames are arranged in a sprite sheet. When an action is performed the proper animation strip is loaded into VRAM. During loading of a strip a ten pointer array is also initialized to the VRAM locations of each frame. Advancing the animation frame only requires incrementing the pointer array iterator.



Example sprite sheet *Note: pink is the transparent color*

5. Software Development Process

I used an incremental build process to develop this project. The lack of detail on required features led me to a trial process for some features. A lack of understanding of how sprites work on the Nintendo DS consumed an enormous amount of time. I had expected a steep learning curve for the DS. Sprites were much harder to learn than I had expected.

5.1: *Sprite Issues*

Sprites are supported in sized up to 64x64 pixels. I had originally attempted to use 128x128 sprites. I spent a couple days trying to get just a single sprite to work at the larger size. Once I learned that 64x64 sprites are the largest size directly supported by the DS I was able to get images on the screen. The first sprites I loaded were scrambled due to incorrect offsets when loading the image into sprite memory. After I learned the details of using sprites progress resumed on a significantly delayed schedule.

6. Results

Through the process of design and implementation I learned the hard way that a design that lacks details leaves too much for interpretation. To continue with this project I would go back to the design phase and work out the requirements in great detail.

6.1: *Project Status*

Champion was only about half finished due to setbacks with sprites. State transitions work correctly and all images load correctly including sprites. Character animations load and animate correctly in the majority of cases. The image processing required to make a sprite sheet takes about an hour and a half per character. Trying to get something to test I only processed sprite sheets for two characters. Both characters currently share a palette which results in interesting colors for the opponent sprite.

6.2: *Future Plans*

I am continuing to work on this project as I would like to use the resulting fight engine in another game I have planned. If I were to release this game I would need to get custom images to prevent a lawsuit from Capcom. In future projects I will be spending more time planning then stick to the plan. Having a detailed plan will make the implementation significantly faster.

7. Conclusion

Working on Champion has been a fun and educating process. I am now able to produce games for the Nintendo DS. I learned that I prefer C++ over Java. I like the power, speed, and flexibility that C++ provides through finer control of memory. I also learned that any games I have planned will require hiring an artist or learning to draw if I want to distribute them.

Choosing a simpler game to implement would have been much easier but I have a tendency to choose ambitious projects. I have not doubt that if I had maintained my focus on this project that I would have completed it. The balancing of classes, family and this project turned out to be rather challenging. Not having a detailed schedule for this project made it easy for me to put it on the back burner when I needed time for other assignments or family time.

Appendix: User Manual

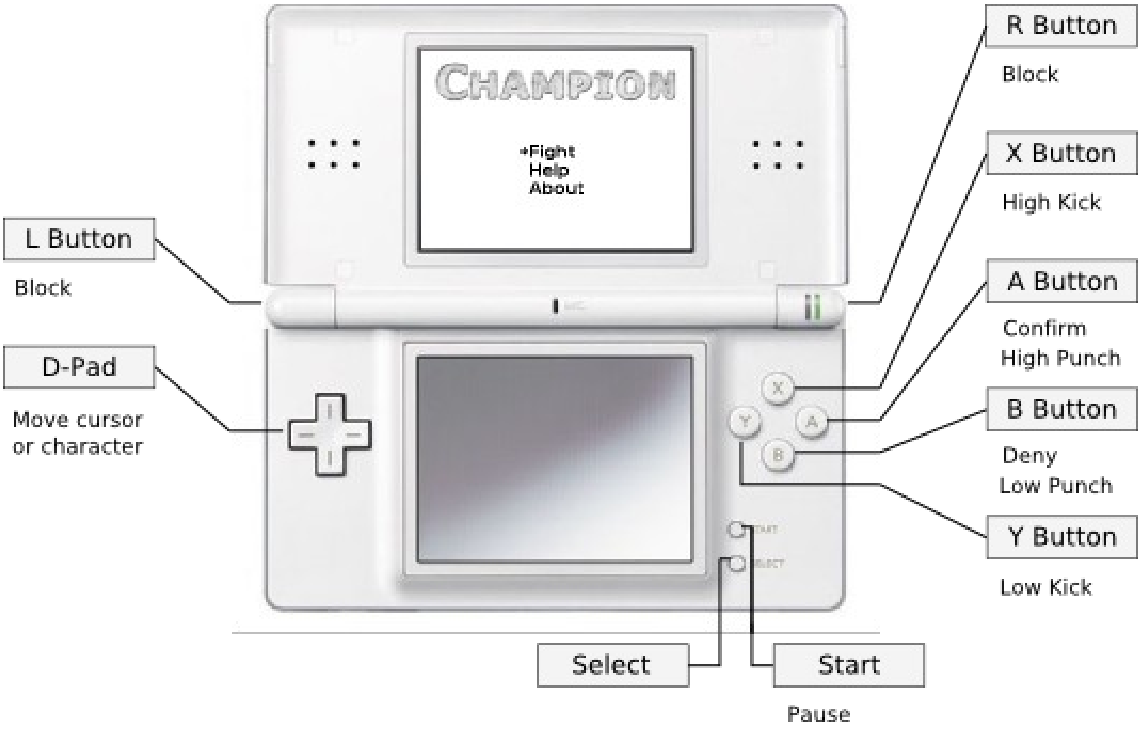
Champion

User Manual

Ian Roskam

Table of Contents:

Controls:

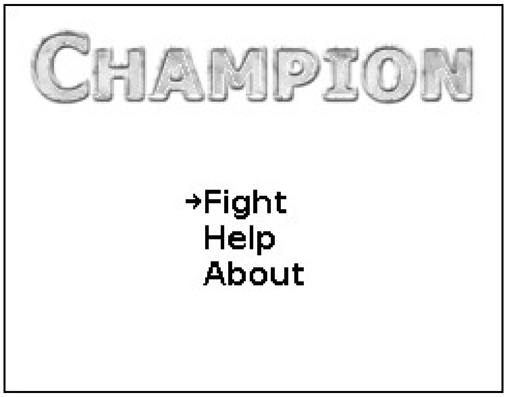


Note: The touch screen is not used for this game.

Getting Started:

Note: At any time outside of a fight match you are able to back out of selections by pressing the 'B' button.

Title Screen:



Starting to fight is just a few selections away.

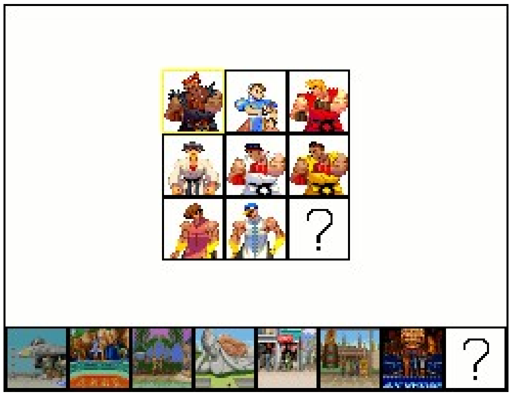
Press 'A' to make selections and 'B' to back out.

Fight: Go to character selection screen.

Help: Display what buttons are used.

About: A little information about Champion.

Character Select Screen:



On this screen you will select your character your opponent and the map you will fight on.

Selecting the '?' will randomly select characters or backgrounds.

Characters:

Akuma, Chun-Li, Ken
Makoto, Ryu, Sean
Yang, Yun

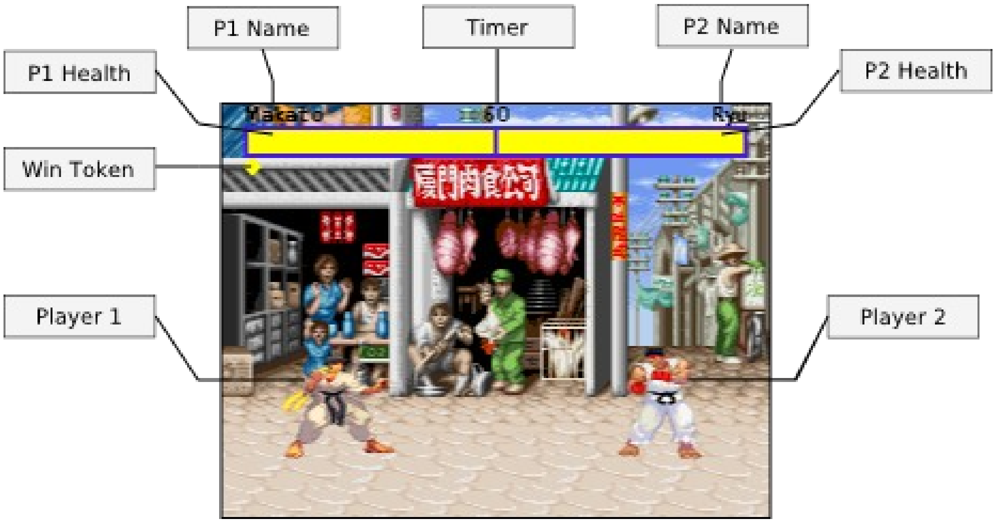
Backgrounds: Airforce, Balcony, Beach, Statue, Street, Temple, Vegas

After making all your decisions pressing 'A' one more time will begin the fight.

Fighting:

Each fight is divided into round. The winner of a fight is the first player to win two rounds. The timer starts at 60 seconds each round. A round is over when the timer runs out or a player's health bar is completely depleted. If the timer expires before a player wins the player with the most health wins. A round will result in a draw if both players have equal health when the timer expires or both players reach zero health at the same time. Win tokens are awarded to the victor at the end of a round. On a draw both players receive a win token.

During a fight the game can be paused by pressing 'Start'. While the game is paused a menu appears giving the option to quit the fight.


P1 Name
Timer
P2 Name
P1 Health
P2 Health
Win Token
Player 1
Player 2
Yakato
60
Ryu