# Specification

This is a fighting game between two fighters, one controlled by the user and the other guided by probabilistic algorithms. The user is to press a key within timed intervals to have his/her fighter perform actions, while the computer utilizes a random number generator and weighted probabilities to have the other fighter perform actions. Actions are executed simultaneously.

There are three selectable fighter types: aggressive, balanced, and defensive. There are three attacks and three corresponding blocks: high, middle, and low. The aggressive fighter has more damage per attack but less defense per block, and has a higher chance of attacking than blocking when controlled by the program. The same idea applies to the defensive fighter.

Each action is implemented as a stance, with each stance having an attack quantity and a defense quantity. Damage is computed by subtracting the defender's stance's defense from the attacker's stance's attack.

The game continues until one fighter's health is depleted, upon which the controller of the other fighter is declared the victor.

# Analysis

```
inputs:  mouse left click for menu navigation
         keyboard q, a, z for high, mid, low blocks
         keyboard w, s, x for high, mid, low attacks
process: have the player select a fighter type
         load initial images and user interface
         run animation
         every update:
             take user input to change fighter's stance
             run probabilistic algorithms to change other fighter's stance
             calculate damage taken and apply to healths
outputs: the score
         a boatload of fun!
```

# Design

Algorithm: Six Windows–created with FLUID/FLTK

- Background selection–User selects a background for the program to display
  - make_window()
    - inputs: none
    - process: execution
    - output: creates a user-defined Fl_Double_Window
  - handle()
    - inputs: mouse; (FL_PUSH, FL_DRAG)
    - processes: check if mouse is inside the selectionBox, if so, enable dragging the selectionBox
    - outputs: none
  - buttonPress()
    - inputs: Fl_Button press
    - processes: saves the user's selected background image in a global variable; hide current window using hide(); show main menu window with show()
    - outputs: main menu appears
- Main menu–Three buttons, user can choose to start game, display info, or exit

  - infoButtonPress()
    * inputs: Fl_Button press
    * processes: hide current window using hide(); show info window with show()
    * outputs: information window appears
  - startButtonPress()
    * inputs: Fl_Button press
    * processes: hide current window using hide(); show selection screen with show()
    * outputs: selection screen appears
  - quitButtonPress()
    * inputs: Fl_Button press
    * processes: close program with std::exit(0)
    * outputs: game ends
- Information screen–Displays game and developer information in two text boxes
  - returnButtonPress()
    * inputs: Fl_Button press
    * processes: hide current window using hide(); show main menu with show()
    * outputs: main menu appears

- Selection Screen–User chooses one of three fighter types: Aggressive - has highest attack and lowest defense; Balanced - has balanced attack and defense statistics; Defensive - has highest defense and lowest attack
  - buttonPress() – one for each fighter type; fighter selection
    - inputs: Fl_Button press
    - processes: display corresponding Fighter image in a Box
    - outputs: image and text description of selected fighter appears
  - startButtonPress()
    - inputs: Fl_Button press
    - processes: initialize selected type of Fighter object; hide current window using hide(); show game window with show()
    - outputs: Fighter object created, game window appears
- Game Screen–User plays the game, display two boxes (characters) for the computer and for the player. The user will press on appropriate keys to attack, defend, or do nothing – each move will either cause the opponent damage or the user to take damage. Each character object has hitpoints and a score as a counter. The score is set at 100 – as each turn progresses, the score decrements by one. When the score is depleted or one of the hitpoints is depleted, the game ends and the game screen closes.
  - handle()

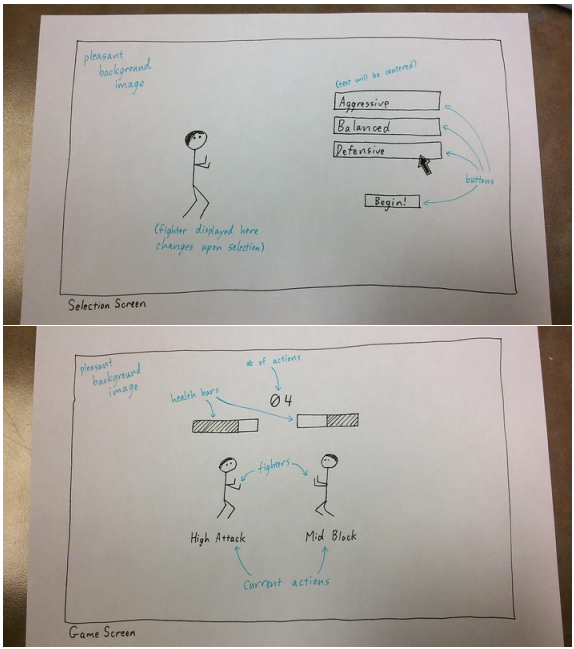
    - inputs: keyboard presses (Q, A, Z, W, S, X, FL_KEYBOARD, FL_KEYPRESS)
    - processes: store keyboard press in a char variable
    - output: image appears
  - runGame()
    - inputs: refer to handle()
    - processes:
      - render assigned image based on key stored from handle()
      - change player's Fighter's stance depending on specific key pressed
      - generate random number with rand() to randomly select opponent$\perp\subseteq\Psi$s move
      - process damage and adjust Fighters' health
      - decrement timer by one
      - check to see if either Fighter's health or the timer is zero
      - utilize Fl::repeat_timeout() to use as a timed, repetitive function
    - outputs: image of appropriate attack and stance appears

- End Screen–Displays "Game Over" along with health, score, and message reading "You Win!" or "You Lose"
  - getScore()
    - inputs: none

    - process: retrieve score and health. Display it. If neither score not health are 0, display "You Win" otherwise, "You Lose!"
    - outputs: display score, health, and game over message
  - buttonPress()
    - inputs: Fl_Button press
    - processes: hide() current window, show() Score Window
    - outputs: score window appears

- Score Screen–
  - doneButtonPress()
    - inputs: Fl_Button press
    - processes: hide current window, show main menu window
    - outputs: main menu appears
  - create fstream object and call fstream functions on it
    - inputs: user enters a name
    - processes: read high scores from a file using ¡fstream¿ functions, called on fstream objects.
      - fileObject.open()
      - isOpen(fileObject, lineString)
      - fileObject.close()
    - outputs: save high scores to the same file as before

# Design (continued)



# Implementation

C++ source code written to file fight.cpp

```
#include <iostream>
#include <sstream>
#include "fight.h"
#include "Fighter.cpp"
```

C++ source code written to file Fighter.cpp

```
#include "Stance.cpp"

class Fighter {
    private:
        Stance* stances[7];
        Stance* currentStance;
        bool blockBonusOn;
        int blockBonus;
        bool attackBonusOn;
        int attackBonus;
        std::string name;
        int chanceOfAttack;
        int health;
    public:
        const static int AGGRESSIVE = 0;
        const static int BALANCED = 1;
        const static int DEFENSIVE = 2;
        const static int AGGRESSIVE_FLIPPED = 3;
        const static int BALANCED_FLIPPED = 4;
        const static int DEFENSIVE_FLIPPED = 5;
```

C++ source code appended to file Fighter.cpp

```
        int getToughness(void);
        int getStanceType(void) { return currentStance->getType(); }
        int getStanceHeight(void) { return currentStance->getHeight(); }
        std::string getStanceName(void) { return currentStance->getName(); }
        std::string getName(void) { return name; }
        void setStance(int type);
        void adjustHealth(int change) { health += change; }
        void toggleBlockBonus(bool set) { blockBonusOn = set; }
        void toggleAttackBonus(bool set) { attackBonusOn = set; }
        Fighter(int style);
};
```

C++ source code appended to file Fighter.cpp

```
Fighter::Fighter(int style) {
    health = 100;
    for (int i = 0; i < 7; i++) {
        stances[i] = new Stance(style, i);
    }
    currentStance = stances[0];
    if (style == 0 || style == 3) {
        blockBonus = 2;
        attackBonus = 3;
        chanceOfAttack = 70;
        name = "Aggressive";
    } else if (style == 1 || style == 4) {
        blockBonus = 4;
        attackBonus = 2;
        chanceOfAttack = 50;
        name = "Balanced";
```

C++ source code appended to file Fighter.cpp

```
    } else if (style == 2 || style == 5) {
        blockBonus = 4;
        attackBonus = 1;
        chanceOfAttack = 40;
        name = "Defensive";
    }
    setStance(0);
    blockBonusOn = false;
    attackBonusOn = false;
}

int Fighter::getPower(void) {
    if (attackBonusOn) {
        return currentStance->getPower() + attackBonus;
        attackBonusOn = false;
    } else {
        return currentStance->getPower();
    }
}
```

C++ source code appended to file Fighter.cpp

```
int Fighter::getToughness(void) {
    if (blockBonusOn) {
        return currentStance->getToughness() + blockBonus;
        blockBonusOn = false;
    } else {
        return currentStance->getToughness();
    }
}

void Fighter::setStance(int type) {
    currentStance = stances[type];
}
```

C++ source code appended to file fight.cpp

```
#include "backgroundselectionwindow.cpp"
```

C++ source code written to file backgroundselectionwindow.cpp

```
int BackgroundSelectionWindow::handle(int event) {
    static bool selectBoxIsGrabbed = false;
    int returnCode = Fl_Double_Window::handle(event);
    if (event == FL_DRAG) {
        returnCode = 1;
        if (selectBoxIsGrabbed) {
            backgroundSelectionWindow->selectBox->
                position(Fl::event_x() - selectBox->w()/2,
            backgroundSelectionWindow->redraw();
        }
    }
```

C++ source code appended to file backgroundselectionwindow.cpp

```
int BackgroundSelectionWindow::getSelectedOption() {
    int x = backgroundSelectionWindow->selectBox->x();
    int y = backgroundSelectionWindow->selectBox->y();
    for (int i = 0; i < 6; i++) {
        if (x > backgroundSelectionWindow->options[i]->x()
                - backgroundSelectionWindow->selectBox->w()
         && x < backgroundSelectionWindow->options[i]->x()
                + backgroundSelectionWindow->options[i]->w()
         && y > backgroundSelectionWindow->options[i]->y()
                - backgroundSelectionWindow->selectBox->h()
         && y < backgroundSelectionWindow->options[i]->y()
                + backgroundSelectionWindow->options[i]->h()) {
            return i;
        }
    }
    return -1;
}
```

C++ source code appended to file backgroundselectionwindow.cpp

```
void BackgroundSelectionWindow::selectButtonPress() {
    int option = backgroundSelectionWindow->getSelectedOption();
    if (option != -1) {
        std::ostringstream fileName;
        fileName << "images/backgrounds/" << option << ".gif";
        backgroundImage = new Fl_GIF_Image(fileName.str().c_str());
        this->hide();
        startWindow->startup();
    }
    std::cout << "Background image set to " << option << std::endl;
}
```

C++ source code appended to file fight.cpp

```
#include "infowindow.cpp"
```

C++ source code written to file infowindow.cpp

```
#include <iostream>
#include <sstream>
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Text_Display.H>

void InfoWindow::backButtonPress() {
    this->hide();
    startWindow->startup();
}

void InfoWindow::startup() {
    this->show();
    backgroundBox->image(backgroundImage);
```

C++ source code appended to file fight.cpp

```
#include "startwindow.cpp"
```

C++ source code appended to file fight.cpp

```
#include "fighterselectionwindow.cpp"
```

C++ source code appended to file fight.cpp

```
#include "gamewindow.cpp"
```

C++ source code appended to file fight.cpp

```
#include "endwindow.cpp"
```

C++ source code appended to file fight.cpp

```
#include "scorewindow.cpp"
```

C++ source code appended to file fight.cpp

Makes pointers to the objects created in Fluid on which to run the game on.

C++ source code appended to file fight.cpp

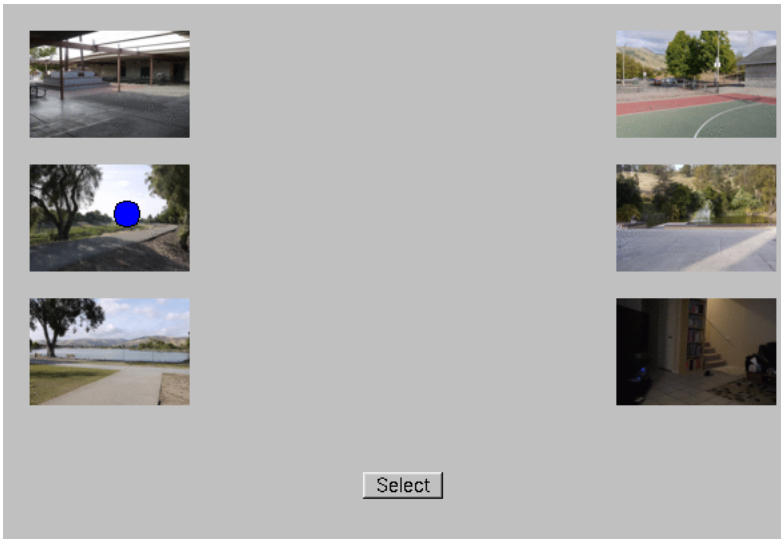
```
BackgroundSelectionWindow* backgroundSelectionWindow;
StartWindow* startWindow;
InfoWindow* infoWindow;
FighterSelectionWindow* fighterSelectionWindow;
GameWindow* gameWindow;
EndWindow* endWindow;
ScoreWindow* scoreWindow;

Fl_GIF_Image* backgroundImage;

```

Makes each window using functions defined in Fluid.

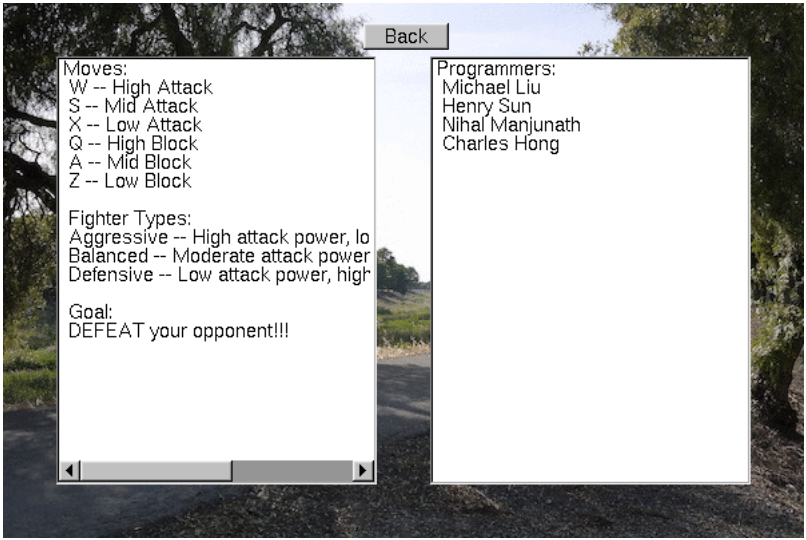
C++ source code appended to file fight.cpp

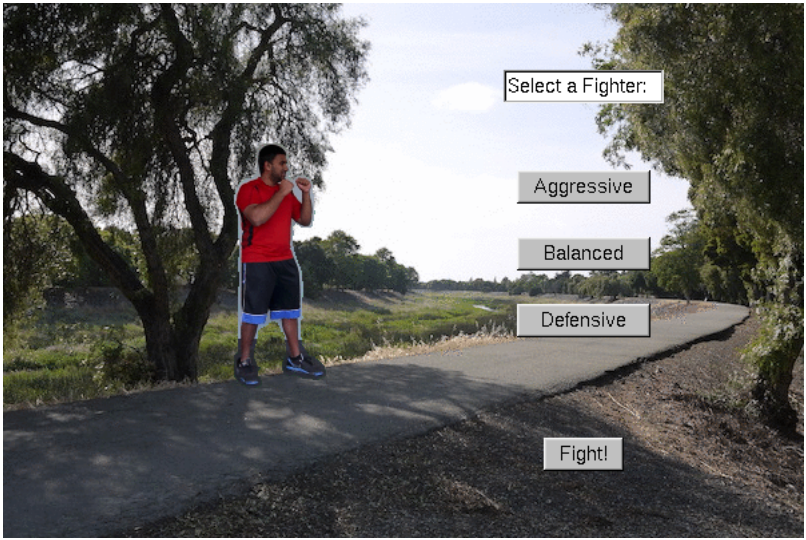
```
int main() {
    backgroundSelectionWindow = makeBackgroundSelectionWindow();
    startWindow = makeStartWindow();
    infoWindow = makeInfoWindow();
    fighterSelectionWindow = makeFighterSelectionWindow();
    gameWindow = makeGameWindow();
    endWindow = makeEndWindow();
    scoreWindow = makeScoreWindow();
    backgroundSelectionWindow->show();
    return Fl::run();
}
```

# Implementation




START
HOW TO PLAY
QUIT


Back
Moves:
W -- High Attack
S -- Mid Attack
X -- Low Attack
Q -- High Block
A -- Mid Block
Z -- Low Block
Fighter Types:
Aggressive -- High attack power, lo
Balanced -- Moderate attack power
Defensive -- Low attack power, high
Goal:
DEFEAT your opponent!!!
Programmers:
Michael Liu
Henry Sun
Nihal Manjunath
Charles Hong


Select a Fighter:
Aggressive
Balanced
Defensive
Fight!


score
95
Balanced
Defensive
health 92
100 health
neutral
high attack


Game Over
Health: 17
Score: 55
High Scores


High Scores
Your Score: 55
Enter your name: Floyd
Submit
1st Name: nihal
1st: 68
2nd Name: Bruce
2nd: 0
3rd Name: Lee
3rd: 0
4th Name: Manny
4th: 0
5th Name: Pacquiao
5th: 0
Back to Menu