# Slot Machine

Soham, Ke, Jagdeep

July 27, 2011

## 1 Specification

Users gamble with Money(Credits) which Casino provides them. Users input the bet amount and then they hit the Spin button to spin the three bars displaying the symbols. The program comes up with a random sequence of predecided symbols. Calculating the amount that a user wins in one spin, it adds that to total credits. The gambling further continues until the user wants to quite or he/she runs out of coins.

## 2 Analysis/Design

Gambler will be provided with 200 credit(coins) initially, and will be given the option to adjust their bet. They can either bet 5 credits, 10 credits, 20 credits, MAX bet (50 credits). Bet amount will be deducted from their total credits as they hit the Spin button.

As the user hits the spin button, 3 symbols will be displayed. The sequence of symbols will thus determine the amount(credit) the user earns in that round.

There would be total of 10 symbols that can be used to display the sequence.

Determining the number of winnig credit.
A User can any time quite to walk away with the credits he/she earned or the program will be terminated as the total credit becomes less then minimum bet.

Managing the Program:

- Initializing the credits to 200 and bet to 0.
- take input from user to (via buttons) to set their bet until user hits Spin button.
- Spin button generates a sequence of 3 symbols and display.

- Read the sequence to determine the credits user won.
- Change the credits accordingly.
- Repeat until user hits the quite (withdraw) button.

## 3 Implementation

"sm.cpp" 2a≡

```
⟨ Include files 12b ⟩
⟨ Constants 13a ⟩
⟨ classes 13b ⟩
⟨ string fill 3 ⟩
⟨ Select image from the data base and put the result into the box 5 ⟩
⟨ Calculate the bet compare with spin result and give out the credits earned 6 ⟩
⟨ Function callback 7c ⟩

int main()
{
 srand(time(0));

⟨ Create main window 2b ⟩
⟨ Create Widgets for input and output and button 4 ⟩
⟨ Register a callback function to be called when button is pressed 7a ⟩
⟨ Show window with its controls 7b ⟩
return(Fl::run());

}
◇
```

Creating main window

⟨ *Create main window* 2b ⟩ ≡

```
Fl_Window* w = new Fl_Window(width,height);
◇
```

Fragment referenced in 2a.

This function is the generator of 3 digits that decide the sequence and display of symbols. It Generates 3 digit unique sequence every time its called and it is used inside the spin function.

⟨ *string fill* 3 ⟩ ≡

```
string fill ()
{
 int f;
 ostringstream ostr;

 f=rand()%1000;
 ostr << setw(3) << setfill('0') << f;
 string sequence=ostr.str();

 return sequence;
}

◇
```

Fragment referenced in 2a.

Need to pass in XY coordinate (upper left corner) of box, and width, height, and label.

⟨ *Create Widgets for input and output and button* 4 ⟩ ≡

```
Widgets widgets;
widgets.credit=200;
widgets.main_bet=0;

w->begin();
widgets.credits = new Fl_Output(100,75,150,30,"CREDITS:");
        ostringstream ostr;
        ostr << widgets.credit;
        widgets.credits->value(ostr.str().c_str());
widgets.bet = new Fl_Output(35,350,100,25,"Bet");
widgets.credit_earned = new Fl_Output(250,350,200,25,"Credits Earned");

widgets.img[0] = new Fl_JPEG_Image("f0.jpg");
widgets.img[1] = new Fl_JPEG_Image("f1.jpg");
widgets.img[2] = new Fl_JPEG_Image("f2.jpg");
widgets.img[3] = new Fl_JPEG_Image("f3.jpg");
widgets.img[4] = new Fl_JPEG_Image("f4.jpg");
widgets.img[5] = new Fl_JPEG_Image("f5.jpg");
widgets.img[6] = new Fl_JPEG_Image("f6.jpg");
widgets.img[7] = new Fl_JPEG_Image("f7.jpg");
widgets.img[8] = new Fl_JPEG_Image("f8.jpg");
widgets.img[9] = new Fl_JPEG_Image("f9.jpg");
widgets.img[10] = new Fl_JPEG_Image("tfp.jpg");
widgets.img[11] = new Fl_JPEG_Image("roc.jpg");

widgets.box1 = new Fl_Box(5,125,150,200);
widgets.box2 = new Fl_Box(158,125,150,200);
widgets.box3 = new Fl_Box(311,125,150,200);

Fl_Button* button_withdraw = new Fl_Button(480,450,160,30,"Withdraw");
Fl_Button* button_spin = new Fl_Button(470,350,130,30,"Spin");
Fl_Button* button_bet_again = new Fl_Button(470,300,130,30,"Bet Again");
Fl_Button* button1 = new Fl_Button(170,400,100,30,"5 Credits");
Fl_Button* button2 = new Fl_Button(270,400,100,30,"10 Credits");
Fl_Button* button3 = new Fl_Button(370,400,100,30,"20 Credits");
Fl_Button* buttonreset = new Fl_Button(50,400,100,30,"Reset");
Fl_Button* buttonmax = new Fl_Button(490,400,100,30,"Bet Max");

w->end();
```

◇

Fragment referenced in 2a.

This function is used to convert a string digist into integer digits and then display corresponding images. The conversion could be done by atoi function, but as these small functions were used in the framework of this program, initially, atoi failed to fit in the program.

⟨ *Select image from the data base and put the result into the box* 5 ⟩ ≡

```
int select_image(char e)
{
int d;
     if (e == '0') {d=0;return d;}
else if (e == '1') {d=1;return d;}
else if (e == '2') {d=2;return d;}
else if (e == '3') {d=3;return d;}
else if (e == '4') {d=4;return d;}
else if (e == '5') {d=5;return d;}
else if (e == '6') {d=6;return d;}
else if (e == '7') {d=7;return d;}
else if (e == '8') {d=8;return d;}
else               {d=9;return d;}

}

void out_image(void* v, string s)
{
Widgets* w = static_cast<Widgets*>(v);

int a=select_image(s[0]);
int b=select_image(s[1]);
int c=select_image(s[2]);
w->box1->image(w->img[a]);
w->box1->redraw();
w->box2->image(w->img[b]);
w->box2->redraw();
w->box3->image(w->img[c]);
w->box3->redraw();
cout << "M here in image";

w->box1->parent()->redraw();
w->box2->parent()->redraw();
w->box3->parent()->redraw();

}

◇
```

Fragment referenced in 2a.

This function determines the credit earned per round considerring the bet for that round.

⟨ *Calculate the bet compare with spin result and give out the credits earned* 6 ⟩ ≡

```
int credit_earned(int c, int bet, string s)
{
char e[4];
double p;
strcpy(e,s.c_str());

s=s.c_str();

if (bet==5){p=0.25;}
else if (bet==10){p=0.50;}
else if (bet==25){p=0.75;}
else if (bet==50){p=1;}

if(s == "111") {c += p*100000000;}
else if(s == "222") {c += p*10000000;}
else if(s == "333") {c += p*5000000;}
else if(s == "444") {c += p*1000000;}
else if(s == "000") {c /= 3;}
else if(e[0] == e[1] and e[1] == e[2]) {c += p*100000;}
else if((e[0] == '1' and e[1] == '1') or (e[0] == '1' and e[2] == '1') or (e[2] == '1' and e[1] =
else if((e[0] == '2' and e[1] == '2') or (e[0] == '2' and e[2] == '2') or (e[2] == '2' and e[1] =
else if((e[0] == '3' and e[1] == '3') or (e[0] == '3' and e[2] == '3') or (e[2] == '3' and e[1] =
else if((e[0] == '4' and e[1] == '4') or (e[0] == '4' and e[2] == '4') or (e[2] == '4' and e[1] =
else if((e[0] == '0' and e[1] == '0') or (e[0] == '0' and e[2] == '0') or (e[2] == '0' and e[1] =
else if(e[0] == e[2] or e[0] == e[1] or e[1] == e[2]) {c += 45;}
else if(e[0] == '1' or e[1] == '1' or e[2] == '1') {c += 25;}
else if(e[0] == '2' or e[1] == '2' or e[2] == '2') {c += 20;}
else if(e[0] == '3' or e[1] == '3' or e[2] == '3') {c += 15;}
else if(e[0] == '4' or e[1] == '4' or e[2] == '4') {c += 10;}
else {}


return c;
}
```

◇

Fragment referenced in 2a.

⟨ *Register a callback function to be called when button is pressed* 7a ⟩ ≡

```
button1->callback(bet_update_1,&widgets);
button2->callback(bet_update_2,&widgets);
button3->callback(bet_update_3,&widgets);
buttonmax->callback(bet_update_max,&widgets);
buttonreset->callback(bet_update_reset,&widgets);
button_bet_again->callback(bet_again,&widgets);
button_spin->callback(spin,&widgets);
button_withdraw->callback(withdraw,&widgets);
◇
```

Fragment referenced in 2a.

⟨ *Show window with its controls* 7b ⟩ ≡

```
w->show();
◇
```

Fragment referenced in 2a.

These are the callback functions in this program. Clicking button triggers certain codes and updates certain variables, changes displays,etc.

⟨ *Function callback* 7c ⟩ ≡

```
⟨ call back function for 5 credit bet button 8a ⟩
⟨ call back function for 10 credit bet button 8b ⟩
⟨ call back function for 20 credit bet button 8c ⟩
⟨ call back function for max credit bet button 9a ⟩
⟨ call back function for reset bet button 9b ⟩
⟨ call back function for bet again button 10 ⟩
⟨ call back function for spin button 11 ⟩
⟨ call back function for withdraw button 12a ⟩
◇
```

Fragment referenced in 2a.

call back function for 5 credit bet button. It updates the bet to 5 credits.

⟨ *call back function for 5 credit bet button* 8a ⟩ ≡

```
void bet_update_1(Fl_Widget* o, void* v)
{
Widgets* w = static_cast<Widgets*>(v);
w->main_bet=5;
 ostringstream ostr;
 ostr << w->main_bet;

w->bet->value(ostr.str().c_str());
}
◇
```

Fragment referenced in 7c.

call back function for 10 credit bet button. It updates the bet to 10 credits.

⟨ *call back function for 10 credit bet button* 8b ⟩ ≡

```
void bet_update_2(Fl_Widget* o, void* v)
{
Widgets* w = static_cast<Widgets*>(v);
w->main_bet=10;
 ostringstream ostr;
 ostr << w->main_bet;

w->bet->value(ostr.str().c_str());

}
◇
```

Fragment referenced in 7c.

call back function for 20 credit bet button. It updates the bet to 20 credits.

⟨ *call back function for 20 credit bet button* 8c ⟩ ≡

```
void bet_update_3(Fl_Widget* o, void* v)
{
Widgets* w = static_cast<Widgets*>(v);
w->main_bet=20;
 ostringstream ostr;
 ostr << w->main_bet;

w->bet->value(ostr.str().c_str());
}
◇
```

Fragment referenced in 7c.

call back function for max credit bet button. It updates the bet to 50 credits.

⟨ *call back function for max credit bet button* 9a ⟩ ≡

```
void bet_update_max(Fl_Widget* o, void* v)
{
Widgets* w = static_cast<Widgets*>(v);
w->main_bet=50;
 ostringstream ostr;
 ostr << w->main_bet;

w->bet->value(ostr.str().c_str());
}
◇
```

Fragment referenced in 7c.

call back function for reseting the bet.

⟨ *call back function for reset bet button* 9b ⟩ ≡

```
void bet_update_reset(Fl_Widget* o, void* v)
{
Widgets* w = static_cast<Widgets*>(v);
w->main_bet=0;
 ostringstream ostr;
 ostr << w->main_bet;

w->bet->value(ostr.str().c_str());
}

◇
```

Fragment referenced in 7c.

call back function for bet again button. It updates the credit earned output to blank and thus giving an idea of starting a new round. It shows spin button and hides itself.

⟨ *call back function for bet again button* 10 ⟩ ≡

```
void bet_again(Fl_Widget* o, void* v)
{
Widgets* w = static_cast<Widgets*>(v);
w->credit_earned->value("");

o->parent()->child(7)->show();
o->parent()->child(8)->hide();


}
```

◇

Fragment referenced in 7c.

call back function for spin button. The main button which generates a sequence of 3 digit ( a string), displays the credit earned in that round by reading the 3 digit string and deciding the appropriate credit earned. It also generates the images corresponding to three digits.
At the end, it hides itself and shows bet again button, thus restricting users to hit the spin button again without reseting the credit earned and the bet.
This button also terminates the program to show that user ran out of credits when credits become lower than bet.

⟨ *call back function for spin button* 11 ⟩ ≡

```
void spin(Fl_Widget* o, void* v)
{
Widgets* w = static_cast<Widgets*>(v);

if(w->credit < 50)
{
        for(int i=0;i<14;i++)
        {
        o->parent()->child(i)->hide();
        }

        o->parent()->child(4)->show();
        w->box2->image(w->img[11]);
        w->box2->redraw();
}

else
{
double difference;

cout << "M here in spin";
string sequence;
sequence=fill();
out_image(v,sequence);
w->credit=w->credit-w->main_bet;
double creditearned=w->credit;
w->credit=credit_earned(w->credit,w->main_bet,sequence);
difference=fabs(w->credit-creditearned);

 ostringstream ostr;
 ostr << difference;

 ostringstream ostr1;
 ostr1 << w->credit;


w->credit_earned->value(ostr.str().c_str());
w->credits->value(ostr1.str().c_str());

o->parent()->child(8)->show();
o->parent()->child(7)->hide();
}

}
◇
```

Fragment referenced in 7c.

call back function for withdraw button. It hides all the buttons and outputs of the window and display an image thanking the users for playing.

⟨ *call back function for withdraw button* 12a ⟩ ≡

```
void withdraw(Fl_Widget* o, void* v)
{
Widgets* w = static_cast<Widgets*>(v);
for(int i=0;i<14;i++)
{
o->parent()->child(i)->hide();
}

o->parent()->child(5)->show();
w->box3->image(w->img[10]);
w->box3->redraw();
}
```

◇

Fragment referenced in 7c.

These are the include files needed for library function calls

⟨ *Include files* 12b ⟩ ≡

```
#include <iostream>
#include<vector>
#include<cmath>
#include<cstdlib>
#include<time.h>
#include<sstream>
#include<iomanip>
#include<fl/fl.h>
#include<fl/fl_window.h>
#include<fl/fl_button.h>
#include<fl/fl_output.h>
#include<fl/fl_jpeg_image.h>
#include<fl/fl_box.h>
using namespace std;
```

◇

Fragment referenced in 2a.

These are the values that will not change during program execution

⟨ *Constants* 13a ⟩ ≡

```
const int total_symbol = 9;
const int symbol=3;
const int width = 640;
const int height = 480;
◇
```

Fragment referenced in 2a.

Classes

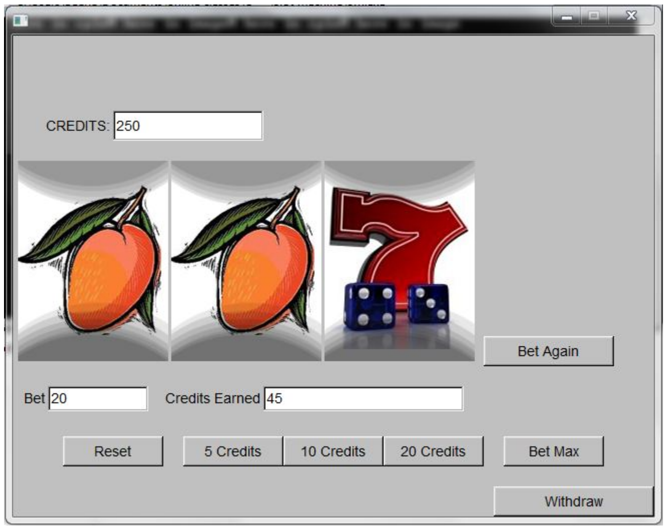
⟨ *classes* 13b ⟩ ≡

```
struct Widgets
{
Fl_Output* out;
Fl_Output* credits;
Fl_Output* credit_earned;
Fl_Output* bet;
Fl_Box* box1;
Fl_Box* box2;
Fl_Box* box3;
Fl_JPEG_Image* img[12];
int main_bet;
int credit;
};

◇
```

Fragment referenced in 2a.

## 4 Test



CREDITS: 250

Bet 20 Credits Earned 45