Slot Machine

Soham

July 25, 2011

1 Specification

User gamble with Money which Casino provides them. User inputs the bet amount and they they hit the deal button to spin the seven bars displaying the symbols. The Machine comes up with a random sequence of predecided symbols. Calculating the amount that a user wins in one spin, it adds that no. of coins to their total coins. 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 100 credit(coins) initially, and will be given the option to adjust the bet. They can either bet 1 coin, 5 coins, 10 coins, or 25 coins to adjust their bets. Bet amount will be deducted from their total credits as they hit the Spin button.

As the user hits the spin button, 5 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.

- If all symbols are alike, then add bet$\times 1000 to the total credits. If no symbol is repeated, then add bet \times 500 to the total credits.$

- If a symbol repeats 4 times, then add betto the total credits.
- If the special sequence appears, then add bet$\times 750 to the total credits. if semi - special sequence appears of four symbols, then add bet to the total credits.$

A User can any time quite to walk away with the credits he/she earned or the ptogram will be terminated as the total credit becomes less then 5.

Managin the Program:

- Initializing the credits to 100 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 7 symbols and display.
- Read the sequence to determine the credits user won.
- Change the credits accordingly.
- Repeat until user hits the quite button.

3 Implementation

"sm.cpp" ?≡

```
⟨ Include files ? ⟩
⟨ Constants ? ⟩
⟨ Function callback ? ⟩
int main()
{

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

}
◇
```

⟨ *Create main window* ? ⟩ ≡

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

Fragment referenced in ?.

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

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

```
Widgets widgets;
w->begin();

widgets.credit = new Fl_Output(100,75,150,30,"CREDITS:");
widgets.bet = new Fl_Output(35,350,100,25,"Bet");
widgets.credit_earned = new Fl_Output(250,350,100,25,"Credits Earned");
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 ?.

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

```
button_withdraw->callback(button_clicked,&widgets);
button_spin->callback(button_clicked,&widgets);
button_bet_again->callback(button_clicked,&widgets);
button1->callback(button_clicked,&widgets);
button2->callback(button_clicked,&widgets);
button3->callback(button_clicked,&widgets);
buttonreset->callback(button_clicked,&widgets);
buttonmax->callback(button_clicked,&widgets);
```

◇

Fragment referenced in ?.

⟨ *Show window with its controls ?* ⟩ ≡

```
w->show();
```

◇

Fragment referenced in ?.

These are the declarations of functions called in this program

⟨ *Function callback ?* ⟩ ≡

```
void button_clicked(Fl_Widget* o, void* v)
{
static int bet_counter=0;
Widgets* w = static_cast<Widgets*>(v);
bet_counter+=5;
w->out->value( bet_counter );
}
```

◇

Fragment referenced in ?.

These are the include files needed for library function calls

⟨ *Include files ?* ⟩ ≡

```
#include <iostream>
using namespace std;
#include<fl/fl.h>
#include<fl/fl_window.h>
#include<fl/fl_button.h>
#include<fl/fl_output.h>
```

◇

Fragment referenced in ?.

These are the values that will not change during program execution

⟨ *Constants ?* ⟩ ≡

```
const int width = 640;
const int height = 480;
struct Widgets
{
Fl_Output* out;
Fl_Output* credit;
Fl_Output* credit_earned;
Fl_Output* bet;
};
```

◇

Fragment referenced in ?.

4 Test