РУЛЕТКА

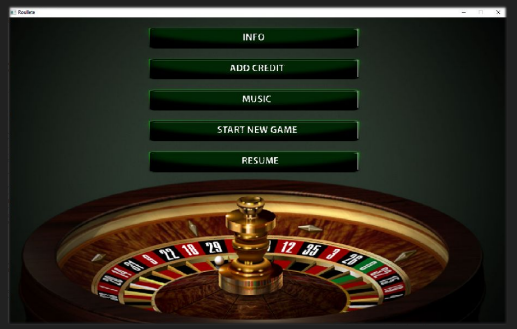
Направена от

Виктор Цонев

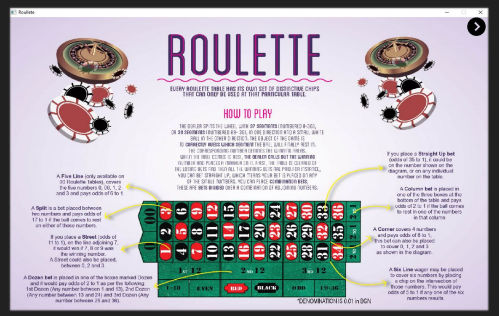
Васил Какевски

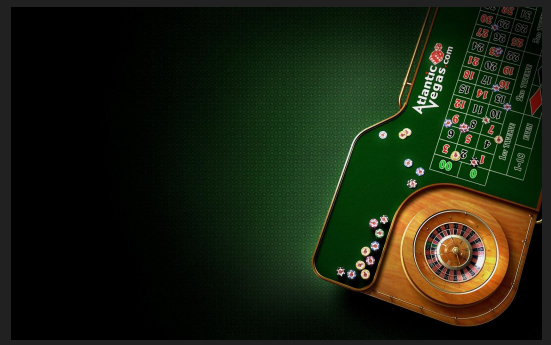
Петър Ефтимов

Росица Маджарова



ЕКРАНИ





Основни екрани в играта

Интро

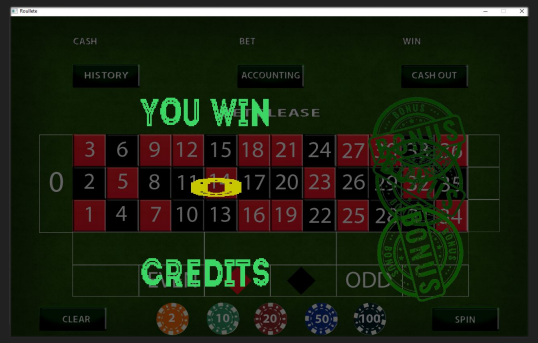
Инфо

Игра

Печалба

Изходен

Бонус



ЕКРАНИ

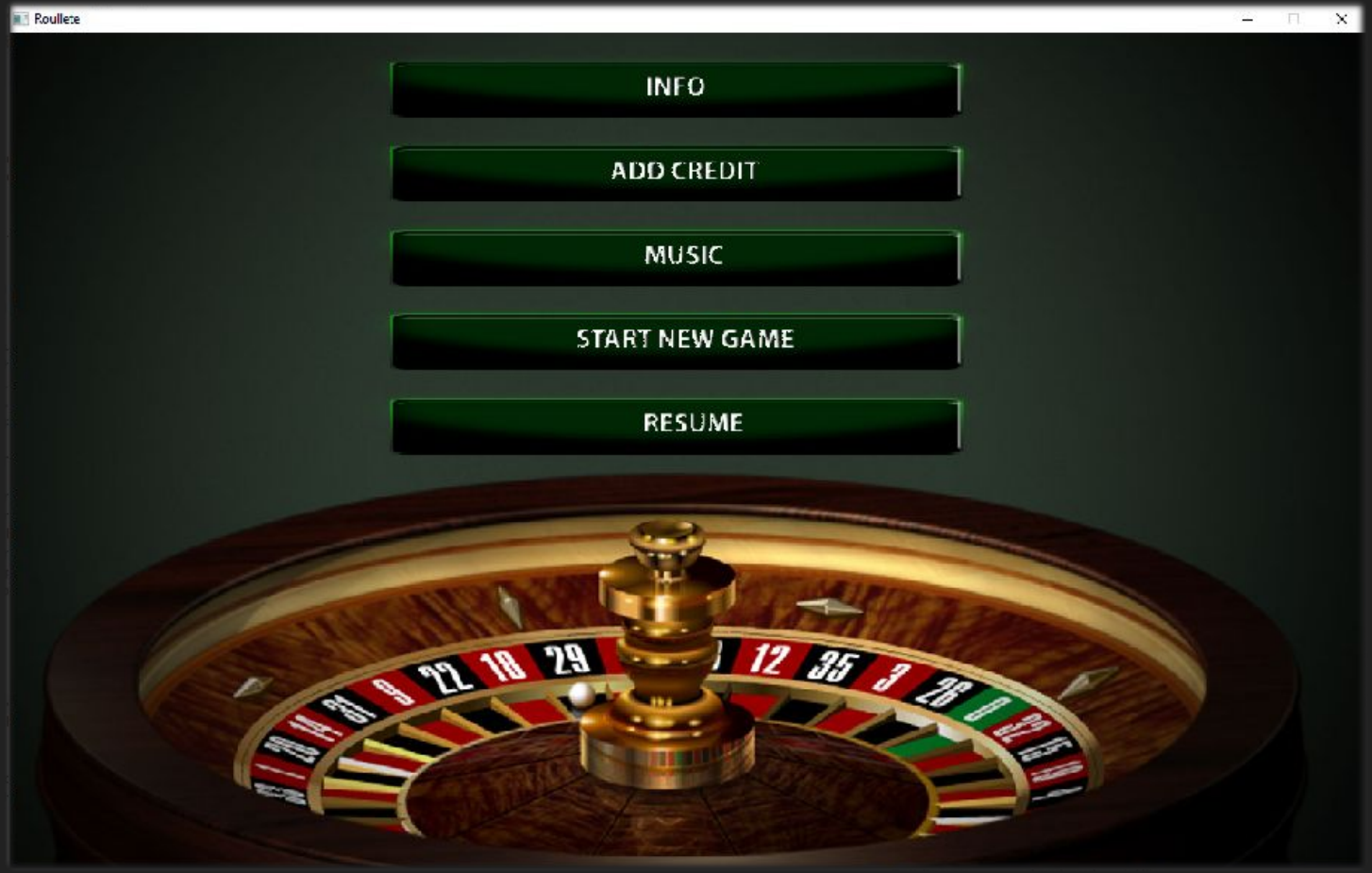
Допълнителни екрани

Въртяща се рулетка

История

Счетоводство

ИНТРО

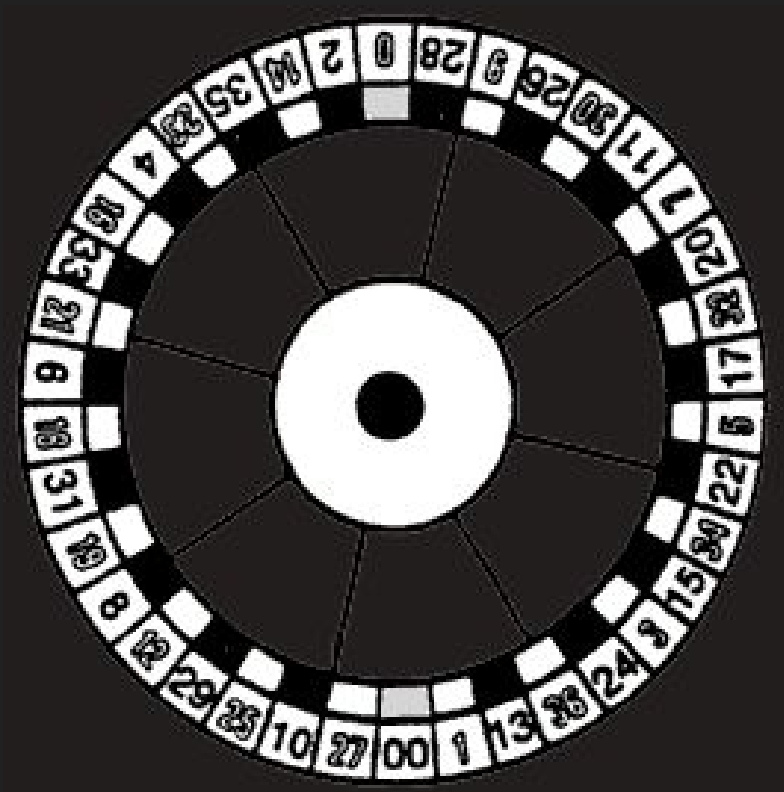


БУТОНИ:

Инфо
Добави кредити
Музика
Старт
Зареди игра

class IntroScreen

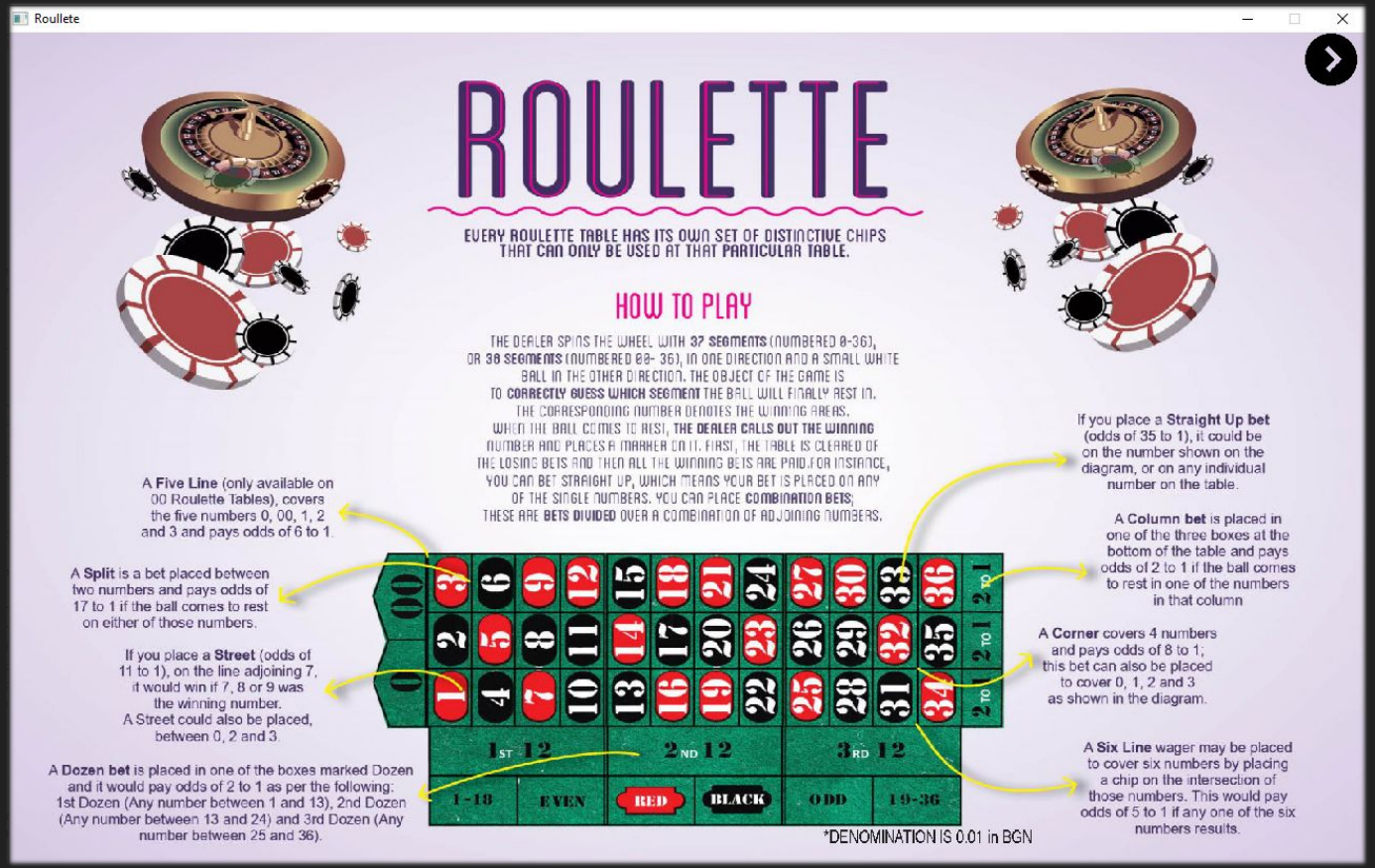
```
class IntroScreen
{
public:
  IntroScreen();
  virtual ~IntroScreen();
  bool Draw();
  bool Clear();
  bool getFlag(){return isActive;}
  void IntroScreenShowCredits(Credits& credits);
public:
  LTexture* introBackground;
  Button* introButtons[INTRO_BUTTONS];
  //bool for mouseEvents
  bool isActive;
};
```



class IntroScreen

```
void
IntroScreen::IntroScreenShowCredits(Credits&
credits)
{
 Clear();
 Draw();
 if (credits.GetCredit())
 {
Text textCredit(SCREEN_W / 2 - 500 / 2 + 50 +
470,
    SCREEN_H / 10 - INTRO_BUTTONS_H / 2
      + (INTRO_BUTTONS_H + 10)
      , 230, 40, 20, "Credits: ", { 100,
      200, 100, 255 });
Text textCreditsNumber(
  SCREEN_W / 2 - 500 / 2 + 50 + 690,
  SCREEN_H / 10 - INTRO_BUTTONS_H / 2
    + (INTRO_BUTTONS_H + 10)
    , 60, 40, 20, credits.GetCredit(),
  { 100, 200, 100 });
 }
}
```

ИНФО



Този екран има само един бутон за Изход

```
class InfoScreen
{
public:
InfoScreen();
 virtual ~InfoScreen();
 bool Draw();
 bool Clear();
 bool getFlag(){return isActive;}
public:
 LTexture* infoBackground;
 Button* infoBack;
 bool isActive;
};
```

class InfoScreen

ИГРАЛЕН ЕКРАН

БУТОНИ:
Завърти рулетката

Изчисти залозите

Изтегли пари

Последните 18

сектора

Счетоводство

class GameBoard

```
class GameBoard
{
public:
 GameBoard();
 virtual ~GameBoard();
 bool Draw();
 bool Clear();
 bool getFlag(){return isActive;}
 void DisplayStatistics(Credits* credits , int
lastWinningNumber);
 int CalcQuadrandClicked(int x, int y);
 void DisplayBets(Credits* credits , int x, int y,
int color,
   bool resume = false);
public:
 LTexture* gameBoard;
 Button* gameBoardPools[POOLS_BUTTON];
 Button* cashOut;
 Button* spin;
 Button* history;
 Button* accounting;
 Button* clearBets;
 Sound* sound;
 bool isActive;
};
```

class GameBoard

```
    if (coordX != -1 && coordY != -1)
   {
    if (credits->GetCredit() >= value[j]) //credits logic
    {
     Button gameBoardPools(coordX, coordY);
     gameBoardPools.loadFromFile("Pools.png");
     gameBoardPools.setWidth(POOLS_W);
     gameBoardPools.setHeight(POOLS_H);
     SDL_Rect rec = { j * 111 + 5, 1, 111, 110 };
     gameBoardPools.render(&rec, 0, NULL,
SDL_FLIP_NONE, 60, 60);
     //while piece to write on
     LTexture overPullUnderText(coordX + 17, coordY + 17);
     overPullUnderText.loadFromFile("BALL.png");
     overPullUnderText.setWidth(POOLS_W);
     overPullUnderText.setHeight(POOLS_H);
     overPullUnderText.render(NULL, 0, NULL,
SDL_FLIP_NONE,
       POOLS_W / 3, POOLS_H / 3);
     sound->play(CLICKBUTTON);
```

```
if (!resume)
     {
      credits->AddBet(value[j]);
      credits->ChangeCredits(-value[j]);
      credits->betByNumberCell[clickedCell] += value[j];
     }
     Text textInPool(coordX + 20, coordY + 22, POOLS_W / 2
+ 8,
       POOLS_H / 2 + 8, 15,
       credits->betByNumberCell[clickedCell]
       , { 0, 0, 0, 255 }, "Intro.otf");
     Text textBet(SCREEN_BOARD_W / 2 + 25, 40, 45, 35,
15,
       credits->GetBet(),
       { 200, 200, 200, 255 });
     recovery->appendToXML(credits->betByNumberCell);
     recovery->appendToXMLCredits(credits->GetCredit(),
       credits->GetBet(), credits->GetCreditsCollected());
    }     //end credits
   }
```

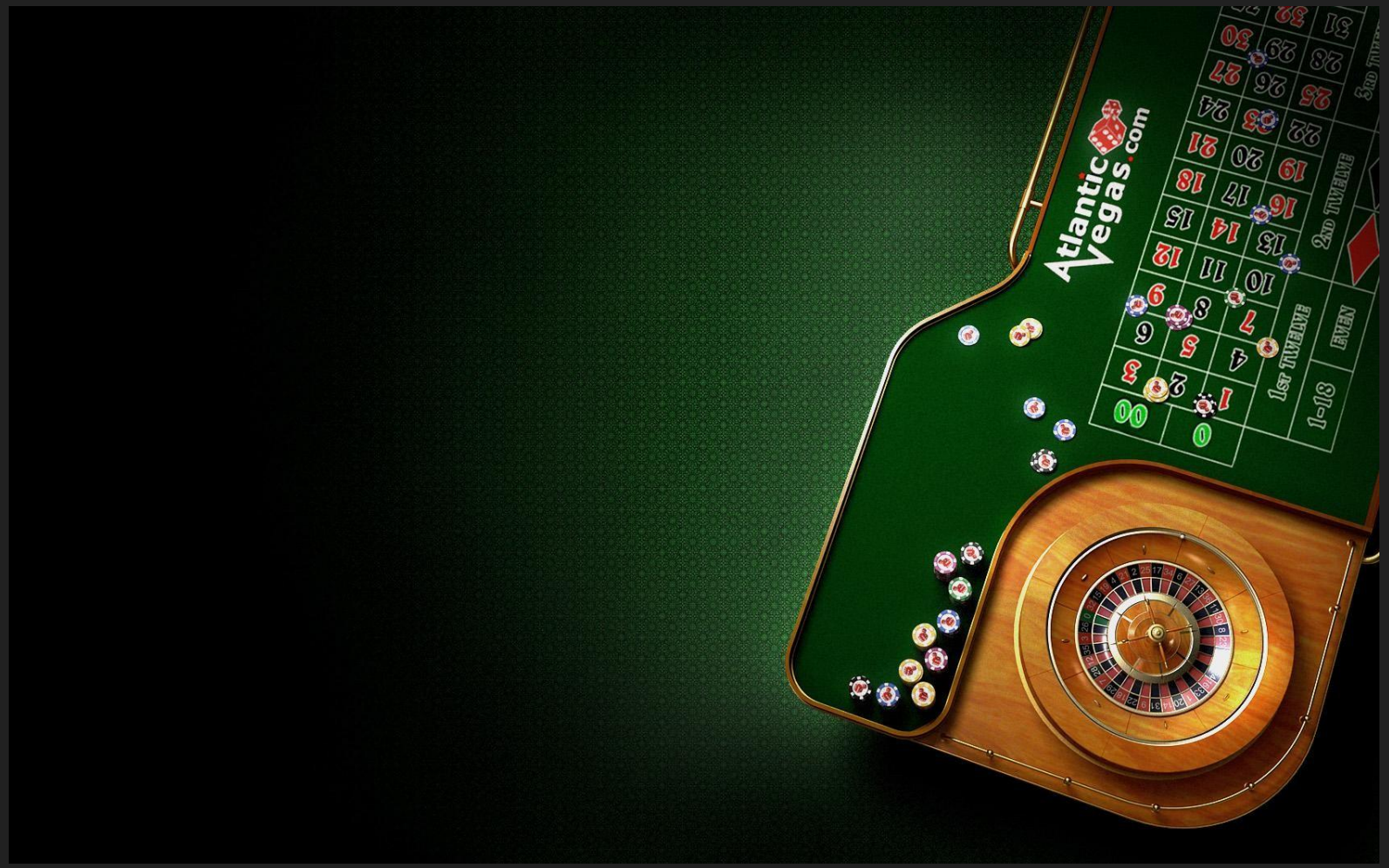
class GameBoard

```
GameBoard::~GameBoard()
{
 for (int i = 0; i < POOLS_BUTTON; i++)
 {
  delete gameBoardPools[i];
 }
 cashOut->free();
 delete cashOut;
 spin->free();
 delete spin;
 history->free();
 delete history;
 accounting->free();
 delete accounting;
 clearBets->free();
 delete clearBets;
 delete sound;
}
```

```
bool GameBoard::Draw()
{
 background->render(NULL, 0, NULL);
isActive = true;
 return true;
}

bool GameBoard::Clear()
{
 SDL_RenderClear(LWindow::gRenderer);
 isActive = false;
 return true;
}
```

ИЗХОДЕН ЕКРАН



Без бутони

6 секунди

презентиране

class OutroScreen

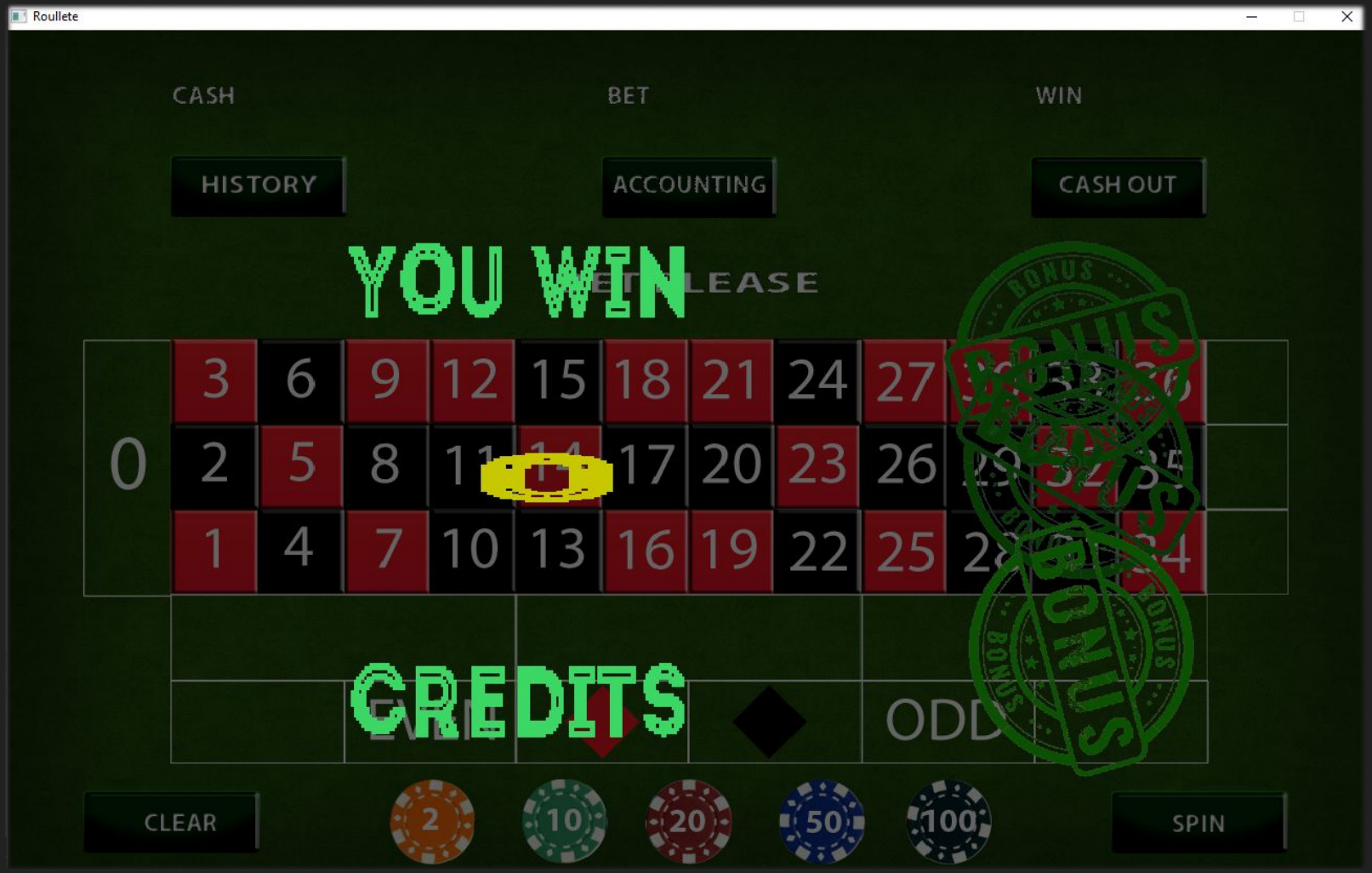
```
class OutroScreen: public Screen
{
public:
 OutroScreen();
 virtual ~OutroScreen();
 bool Draw();
 bool Clear();
 void Show(Credits* credits);
};
```

```
OutroScreen::OutroScreen()
  : Screen()
{
 background->loadFromFile("OutroScreen2.jpg");
 background->setWidth(SCREEN_W);
 background->setHeight(SCREEN_H);
 isActive = false;
}
OutroScreen::~OutroScreen()
{
}
```

class OutroScreen


```
void OutroScreen::Show(Credits* credits)
{
 double scale = 0.6;
 Text textMoneyNumber((SCREEN_W / 8 +
520) * scale, 400 * scale, 180 * scale,
   200 * scale, 15, credits->GetCredit() *
DENOMINATION, { 200, 10, 10,
     255 });
 cout << credits->GetCredit() << "bbbb" <<
endl;
}
bool OutroScreen::Clear()
{
 SDL_RenderClear(LWindow::gRenderer);
 isActive = false;
 return isActive;
}
```

БОНУС ЕКРАН



Без бутони

3 секунди

презентиране

анимация с

бонус

печати.

class BonusScreen

```
class BonusScreen: public Screen
{
public:
 BonusScreen();
 virtual ~BonusScreen();
 bool Draw();
 bool Clear();
 bool getBonusCreditsText(Credits*);
public:
 LTexture* bonusSticker;
 Sound* sound;
};
```

class BonusScreen

```
bool BonusScreen::Draw()
{
 background->render(NULL, 0, NULL);
 SDL_Delay(2000);
 sound->play(WIN);
 srand(time(0));
 SDL_Color col = {
   rand() % 200 + 50,
   rand() % 200 + 50,
   rand() % 200 + 50,
   255 };
```

```
Text textYouWin(SCREEN_W / 4, SCREEN_H / 4,
SCREEN_W / 4,
   SCREEN_H / 8, 30, "You Win", col, "Intro Inline.otf");
 Text textCredits(SCREEN_W / 4, SCREEN_H * 3 / 4,
   SCREEN_W / 4, SCREEN_H / 8,
   30, "credits", col, "Intro Inline.otf");
 for (int i = 0; i < 3; i++)
 {
  SDL_SetTextureColorMod(bonusSticker->getTexture(),
    rand() % 100 + 150, rand() % 70 + 70, rand() % 100 + 150);
  bonusSticker->setY((i + 1) * SCREEN_H / 6 + rand() % 50);
  bonusSticker->render(NULL, i * 45, NULL);
  SDL_Delay(500);
 }
 isActive = false;
 return true;
}
```

class BonusScreen

```
bool BonusScreen::getBonusCreditsText(Credits* credits)
{
 srand(time(0));
 Text textWinnings(SCREEN_W / 3,
 SCREEN_H * 2 / 4,
   SCREEN_W / 8, SCREEN_H / 12, 40,
   credits->GetCreditsCollected(),
   { 200, 200, 0 }, "Intro Inline.otf");
 SDL_Delay(1500);
 return true;
}
```

ВЪРТЯЩА СЕ РУЛЕТКА

Допълнителен екран включващ анимация на въртящо се колело с топче

class SpinScreen

```
class SpinScreen: public Screen
{
public:
 SpinScreen();
 virtual ~SpinScreen();
 bool Draw();
 bool Clear();
 int GenerateWinningNumber();
 int GetWinningNumber();
 bool IsReadyForBonus();
 private:
 LTexture* roulette;
 LTexture* wheel;
 LTexture* ball;
 Sound* sound;
 private:
 int numberOfSpins;
 int winningNumber;
 void FillTheMapsOfRoulette();
};
```

```
bool SpinScreen::Draw() {
 int result = GenerateWinningNumber();
 background->render(NULL, 0, NULL);
 sound->play(SPINROULETTE);
 SDL_Delay(2000);
 int mFrame = 0;
 double angleWheel = -3 + 9.7 *
(posissionToNumberInRoulette[result] - 5);
 double stepWheel = 2;
 double maxR = 310, minR = 210,
   currentR = maxR, minAngle = -95,
   step = M_PI / 36, angleBall = 0;
 do {
  angleBall -= (maxR - currentR) / (12 * 200.0 / M_PI);
  currentR -= (maxR - minR) / (10 * 200.0 / M_PI);
  angleBall -= step;
  ball->setX(SCREEN_W / 2 - BALL_W / 2 + cos(angleBall) *
maxR);
  ball->setY(SCREEN_H / 2 - BALL_H / 2 + sin(angleBall) *
maxR);
if (mFrame % 3 == 0)
  {
   roulette->render(NULL, 0);
   wheel->render( NULL, angleWheel);
   ball->render( NULL, 0);
  }
  angleWheel += stepWheel;
  maxR -= 0.1;
  mFrame++;
 }
 while (angleBall > minAngle);
 sound->play(WINING_NUMBER);
 SDL_Delay(2000);
 sound->music(result);
 SDL_Delay(1000);
 numberOfSpins++;
 return true;
}
```

ИСТОРИЯ

Допълнителен екран показващ последните 18 печеливши числа.

СЧЕТОВОДСТВО

Допълнителен екран показващ статистики от играта.

ЕКРАН ПРИ ПЕЧАЛБА

Допълнителен екран без бутони
6 секунди презентиране анимация с въртящи се монети

class WinScreen

```
class WinScreen: public Screen
{
public:
 WinScreen();
 virtual ~WinScreen();
 bool Draw();
 bool Clear();
 void WinAnimation();
 void ShowCredits(Credits*);
private:
 void fillRectPosition();
 LTexture* coin[COIN_COUNT];
};
```

```
void WinScreen::ShowCredits(Credits* credits)
{
 WinAnimation();
 SDL_Color color { rand() % 255, rand() % 255, rand() % 255, rand()
% 255 };
 Text winAmmount(SCREEN_W / 2 - 300 / 2, SCREEN_H * 3 / 5,
   300, 100, 20, credits->GetWinProfit(), color);
 SDL_Delay(1000);
}
```

class WinScreen

```
void WinScreen::ShowCredits(Credits*
credits)
{
 WinAnimation();
 SDL_Color color { rand() % 255, rand()
% 255, rand() % 255, rand() % 255 };
 Text winAmmount(SCREEN_W / 2 -
300 / 2, SCREEN_H * 3 / 5,
   300, 100, 20, credits->GetWinProfit(),
color);
 SDL_Delay(1000);
}
```

```
void WinScreen::WinAnimation()
{
 vector<SDL_Rect> goldCoins;
 for (int i = 0; i < 10; i++)
  goldCoins.push_back(
    { COIN_W * i, 0, COIN_W, COIN_H });
 for (int i = 0; i < (int) coinFlipz.size(); i++) {
  coin[i]->setX(coinFlipz[i].x);
  coin[i]->setY(coinFlipz[i].y);
  SDL_Color color { rand() % 255, rand() % 255, rand() % 255,
rand() % 255 };
  Text winText(SCREEN_W / 2 - 300 / 2, SCREEN_H / 2 - 10/2,
    300, 100, 20, "YOU WIN", color);
  for (int j = 0; j < 10; j++)  {
   SDL_RenderCopyEx(LWindow::gRenderer,
coin[i]->getTexture(),
     &goldCoins[j], &coinFlipz[i], -90, NULL, SDL_FLIP_NONE);
   SDL_RenderPresent(LWindow::gRenderer);
   SDL_Delay(5);
  } } }
```

```
void WinScreen::fillRectPosition()
{
 SDL_Rect rec = { 20, 675, 90, 90 };
 coinFlipz.push_back(rec);
 rec.x = 128;
 rec.y = 690;
 rec.w = 95;
 rec.h = 95;
 coinFlipz.push_back(rec);
 rec.x = 230;
 rec.y = 650;
 coinFlipz.push_back(rec);
 rec.x = 358;
 rec.y = 580;
 coinFlipz.push_back(rec);
 rec.x = 485;
 rec.y = 650;
 rec.w = 105;
 rec.h = 105;
 coinFlipz.push_back(rec);
 rec.x = 270;
 rec.y = 445;
 rec.w = 95;
 rec.h = 95;
 coinFlipz.push_back(rec);
 rec.x = 340; rec.y = 490;
 rec.w = 90; rec.h = 90;
 coinFlipz.push_back(rec);
 rec.x = 200; rec.y = 575;
 rec.w = 75; rec.h = 75;
 coinFlipz.push_back(rec);
 rec.x = 120; rec.y = 635;
 rec.w = 65; rec.h = 65;
 coinFlipz.push_back(rec);
 for (int i = 0; i < 9; i++){
  rec = coinFlipz[i];
  rec.x = SCREEN_W - rec.x - rec.w + 15;
  coinFlipz.push_back(rec); } }
```

ВЪЗСТАНОВЯВАНЕ НА ИГРАТА

При натискане на бутон RESUME в началният екран играта се възстановява до момента на прекъсване. Възстановяването се реализира с помощта на XML файлове.


```
if (intro.introButtons[4]->isClicked(&e))
    {
     intro.sound->play(CLICKBUTTON);
     intro.Clear();
     if (game.Draw())
     {
      credits = recovery.readXMLWriteCredit(
        "roulette_recovery_credits.xml");
      credits.betByNumberCell = recovery.readXMLWriteMap(
        "roulette_recovery.xml");
      game.DisplayStatistics(&credits,
        recovery.lastWiningNumbers.back());
      for (int y = 285; y < SCREEN_BOARD_H; y += 75)
       for (int x = 77;
         x < SCREEN_BOARD_W - 77; x += 75)
       {
        if (y > 550)
         x += 75;
        if (credits.betByNumberCell[
          game.CalcQuadrandClicked(x, y)] > 0)
         game.DisplayBets(&credits, x, y, 1,
           true, &recovery);
       }
      }
     }
```

ДЕМО

БЛАГОДАРИМ ЗА ВНИМАНИЕТО