РУЛЕТКА

Направена от

Виктор Цонев

Васил Какевски

Петър Ефтимов

Росица Маджарова

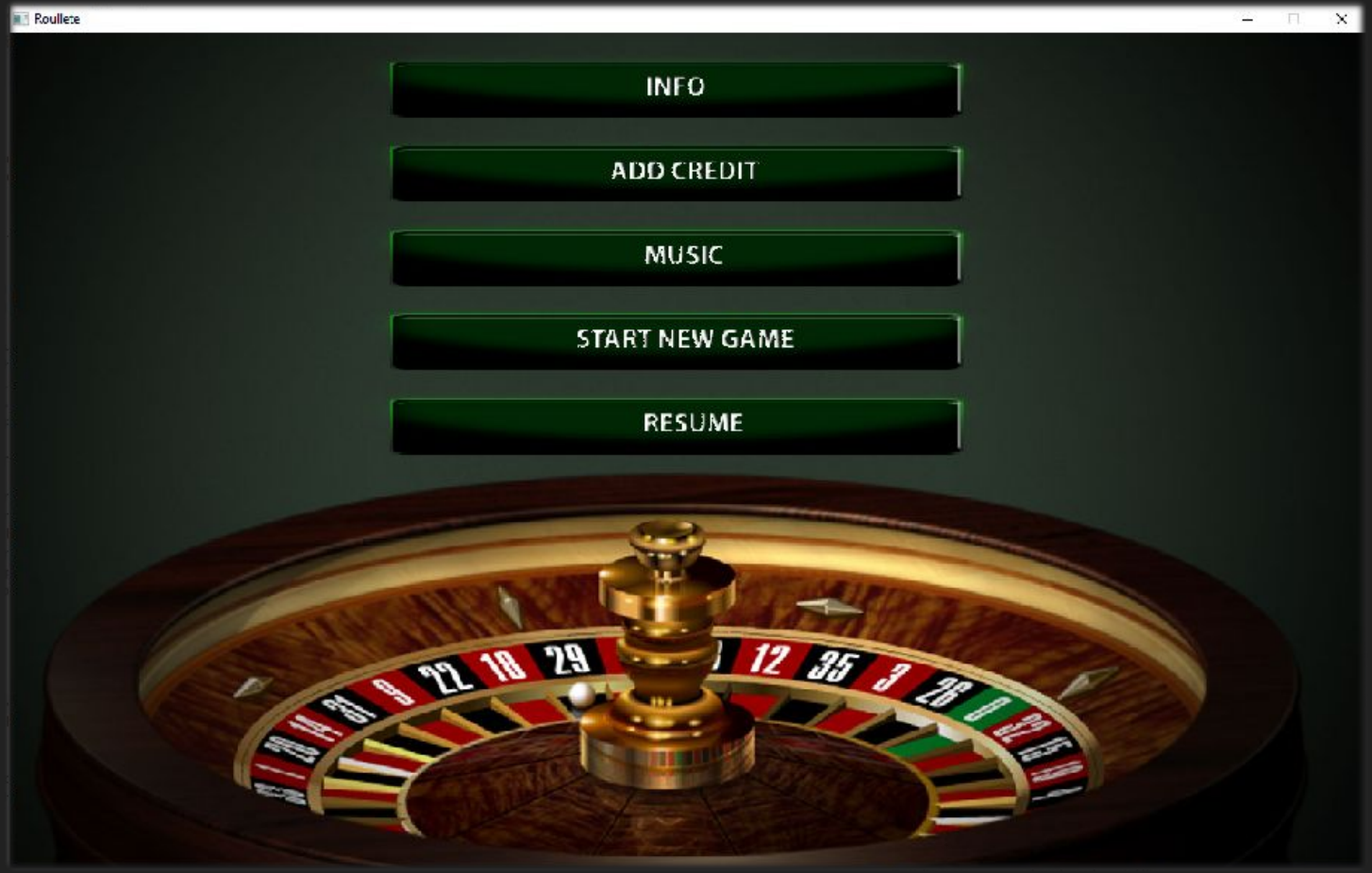
ЕКРАНИ

Основните екрани в играта са Интро, Инфо, Игра, Печалба, Изходен и Бонус.

Допълнителните екрани са Въртяща се рулетка, История, Счетоводство и

екран при печалба.

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;
};
```

```
IntroScreen::IntroScreen()
{
 introBackground = new LTexture(0,0);

introBackground->loadFromFile("IntroBackgrou
ndFinal.bmp");
 introBackground->setWidth(SCREEN_W);
 introBackground->setHeight(SCREEN_H);
 for (int i = 0 ; i < INTRO_BUTTONS ; i++)
 {
  introButtons[i]= new Button (  SCREEN_W / 2
- INTRO_BUTTONS_W / 2,
    SCREEN_H / 13 - INTRO_BUTTONS_H / 2
+ i * (INTRO_BUTTONS_H + 30));
 introButtons[i]->loadFromFile("1.png");
 introButtons[i]->setWidth(580);
 introButtons[i]->setHeight(50);
 }
 isActive=false;
}
```

class IntroScreen

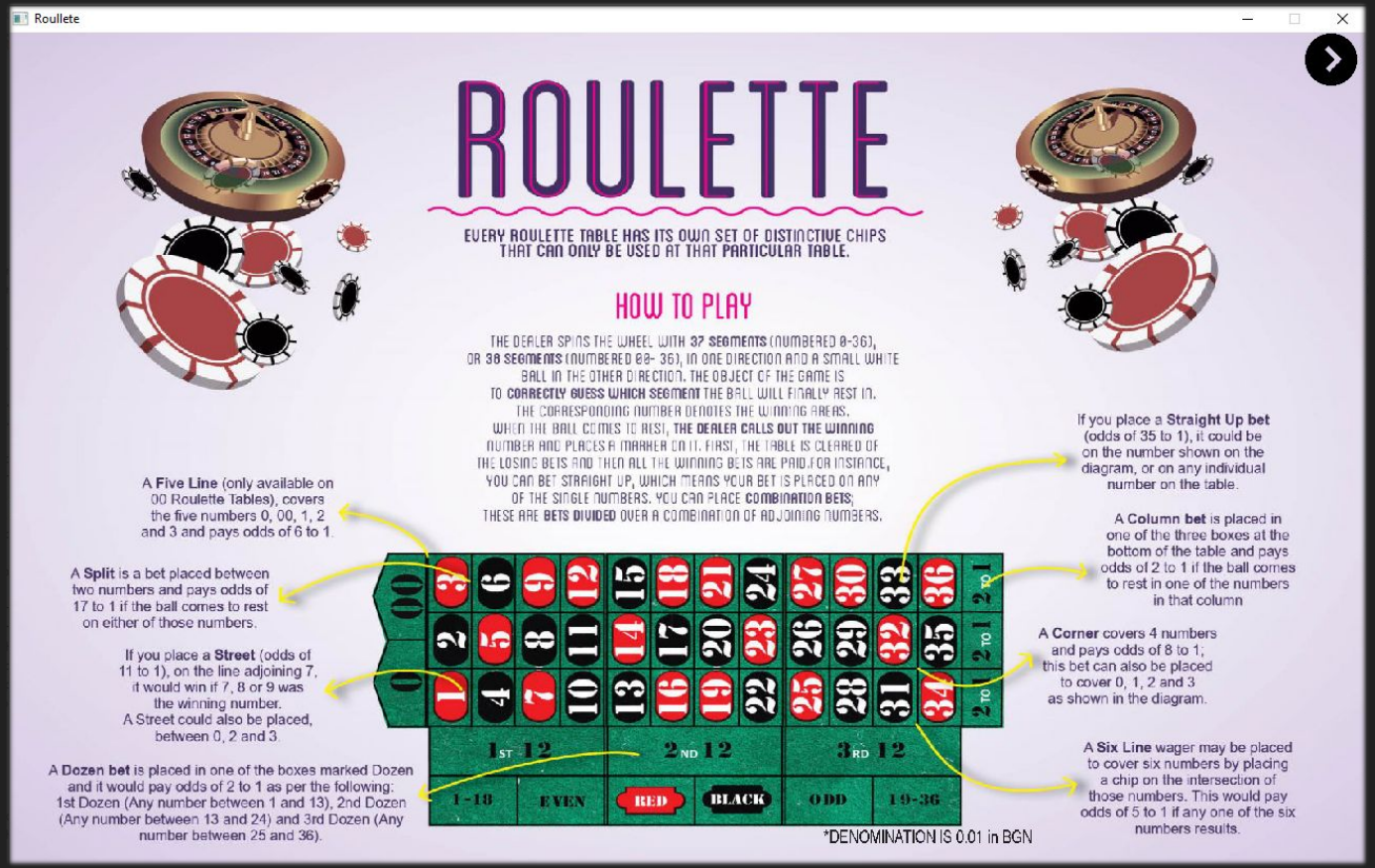
```
IntroScreen::~IntroScreen()
{
 introBackground->free();
 delete introBackground;
 introBackground = NULL;
 for(int i = 0 ; i < INTRO_BUTTONS ; i++)
 {
  introButtons[i]->free();
  delete introButtons[i];
 }
}
```

```
bool IntroScreen::Draw()
{
if(introBackground->render(NULL,0,NULL))
{
 isActive = true;
}
return true;
}
bool IntroScreen::Clear()
{
SDL_RenderClear(LWindow::gRenderer);
isActive=false;
return true;
}
```

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

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

```
InfoScreen::InfoScreen()
{
 infoBackground= new LTexture (0,0);

infoBackground->loadFromFile("rouletterules.jp
g");
 infoBackground->setWidth(SCREEN_W);
 infoBackground->setHeight(SCREEN_H);
 infoBack = new Button (SCREEN_W -
INFO_BUTTON_W - 10, 0);
 infoBack->loadFromFile("BackButton.png");
 infoBack->setWidth(INFO_BUTTON_W);
 infoBack->setHeight(INFO_BUTTON_H);
 isActive = false;
}
```

```
bool InfoScreen::Draw()
{
 if(infoBackground->render(NULL,0)
  && infoBack->render(NULL,180))
 {
  Text textDenomination(SCREEN_W * 3 / 5,
SCREEN_H - 40, 200, 20, 30,
    "*DENOMINATION IS 0.01 in BGN", { 30,
30, 30, 255 });
  isActive=true;
 }
 return true;
}
```

```
InfoScreen::~InfoScreen()
{
 infoBackground->free();
 delete infoBackground;
 infoBack->free();
 delete infoBack;
}

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

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

```
GameBoard::GameBoard()
{
 double scaleX = 1200 / 1300.0;
 double scaleY = 750 / 800.0;
 gameBoard = new LTexture(0,0);

gameBoard->loadFromFile("EuropeanRouletteFinal.bmp
");
 gameBoard->setWidth(1200);
 gameBoard->setHeight(750);
 cashOut = new Button (958 * scaleX , 120 * scaleY);
 cashOut->loadFromFile("1.png");
 cashOut->setWidth(GAME_BOARD_BUTTON_W);
 cashOut->setHeight(GAME_BOARD_BUTTON_H);
 spin = new Button(1035 * scaleX , 725 * scaleY);
 spin->loadFromFile("1.png");
 spin->setWidth(GAME_BOARD_BUTTON_W);
 spin->setHeight(GAME_BOARD_BUTTON_H);
history = new Button(150 * scaleX, 120 * scaleY);
 history->loadFromFile("1.png");
 history->setWidth(GAME_BOARD_BUTTON_W);
 history->setHeight(GAME_BOARD_BUTTON_H);
 accounting = new Button (555 * scaleX , 120 * scaleY);
 accounting->loadFromFile("1.png");
 accounting->setWidth(GAME_BOARD_BUTTON_W);
 accounting->setHeight(GAME_BOARD_BUTTON_H);
 clearBets = new Button(70 * scaleX , 725 * scaleY);
 clearBets->loadFromFile("1.png");
 clearBets->setWidth(GAME_BOARD_BUTTON_W);
 clearBets->setHeight(GAME_BOARD_BUTTON_H);
 for(int i = 0 ; i < POOLS_BUTTON ; i++)
  {
   gameBoardPools[i]=new Button (335 + i * 123  *
scaleX, 710 * scaleY);
   gameBoardPools[i]->loadFromFile("Pools.png");
   gameBoardPools[i]->setHeight(4 * POOLS_W);
   gameBoardPools[i]->setWidth(4 * POOLS_H);
  }
 isActive = false;
 sound= new Sound;
 sound->load();
}
```

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;
}
```

class GameBoard

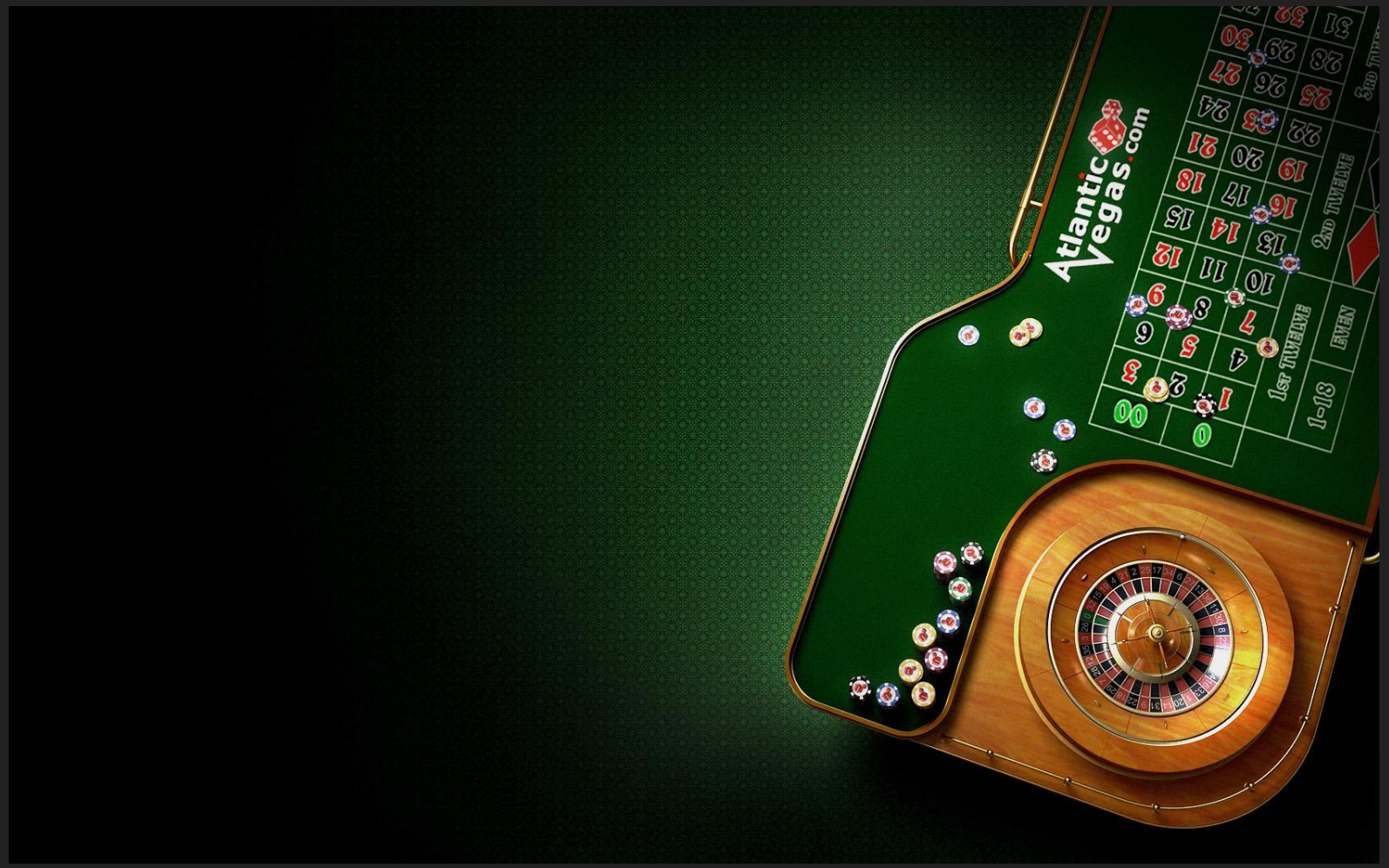
```
int GameBoard::CalcQuadrandClicked(int x, int y)
{
/ int sequence = -1;
 int clickedCell = -1;

 for (int line = 0; line < 3; line++)
 {
  for (int i = 0; i < 13; i++)
  {
   sequence++;
   if (x >= 76 + (75 * i)
     && x <= (151 + 75 * i)
     && y >= 280 + (75 * line)
     && y <= 350 + (75 * line))
   {
    clickedCell = sequence;
   }
  }
 }
```

```
for (i nt i = 0; i < 4; i++)
 {
  if (x >= 300 + (150 * i) && x <= 450 + (150 * i)
&& y >= 585
    && y <= 650)
  {
   clickedCell = 39 + i;
  }
 }
// cout << "clickedCell:" << clickedCell << endl;
 return clickedCell;
}
```

class GameBoard

void GameBoard::DisplayBets(Credits* credits, int x, int y, int color, bool resume)

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);
};
```

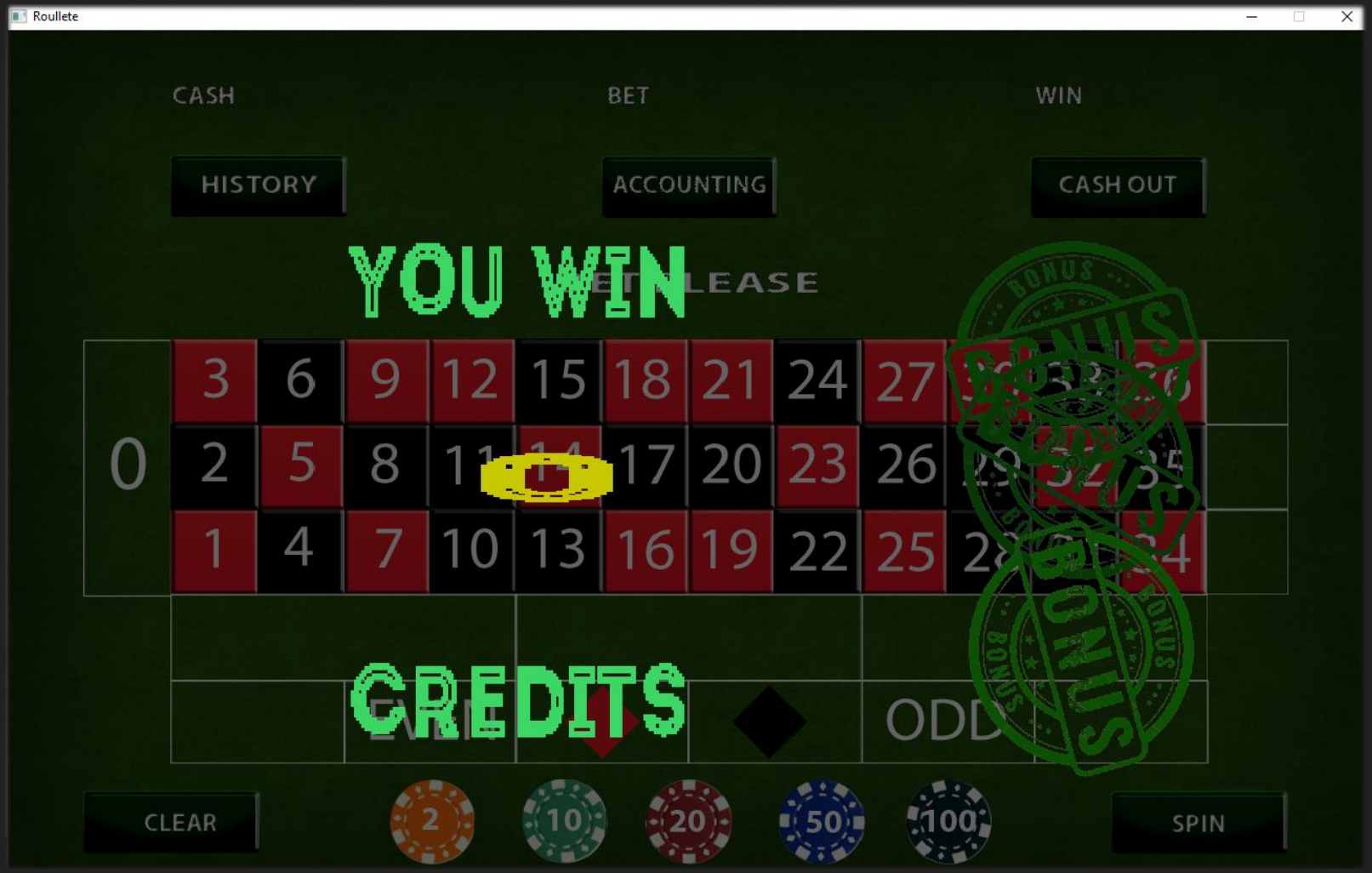
```
OutroScreen::OutroScreen()
  : Screen()
{
 background->loadFromFile("OutroScreen2.jpg");
 background->setWidth(SCREEN_W);
 background->setHeight(SCREEN_H);
 isActive = false;
}
OutroScreen::~OutroScreen()
{
}
```

class OutroScreen

```
bool OutroScreen::Draw()
{
 double scale = 0.6;
 if (background->render(NULL, 0))
 {
  Text textMoney((SCREEN_W / 8 + 520 + 180)
* scale, 400 * scale,
    220 * scale,
    200 * scale, 15, "BGN", { 255, 255, 255 });
  Text textThankYou(SCREEN_W / 8 * scale,
30 * scale, 900 * scale,
    200 * scale, 20, "THANK YOU FOR
PLAYING", { 0, 200, 0, 155 });
  Text textYouHave(SCREEN_W / 8 * scale,
400 * scale, 500 * scale,
    200 * scale, 15, "You have", { 255, 255, 255
});
  return true;
 }
 return false; }
```

```
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;
};
```

```
BonusScreen::BonusScreen()
  : Screen()
{
 background->loadFromFile("EuropeanRouletteFinal.bmp");
 background->setWidth(SCREEN_W);
 background->setHeight(SCREEN_H);
 background->setAlpha(100);
 isActive = false;
 bonusSticker = new LTexture(SCREEN_W * 4 / 6,
SCREEN_H / 6);
 bonusSticker->loadFromFile("Lucky.png");
 bonusSticker->setWidth(300);
 bonusSticker->setHeight(300);
 sound = new Sound;
}
BonusScreen::~BonusScreen()
{
 delete 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);
 }
 //always false because we dont have active screen
 isActive = false;
 return true;
}
```

class BonusScreen

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

class SpinScreen

class SpinScreen

```
SpinScreen::SpinScreen() : Screen()
{
 numberOfSpins = 1;
 background->loadFromFile("EuropeanRouletteFinal.bmp");
 background->setWidth(SCREEN_W);
 background->setHeight(SCREEN_H);
 background->setAlpha(100);
 //center wheel boarder
 roulette = new LTexture(250, 0);
 roulette->loadFromFile("RouletteBoard.png");
 roulette->setWidth(SCREEN_H);
 roulette->setHeight(SCREEN_H);
 wheel = new LTexture(SCREEN_W / 2 - WHEEL_W / 2,
   SCREEN_H / 2 - WHEEL_H / 2);
 wheel->loadFromFile("wheel2.png");
  wheel->setWidth(WHEEL_W);
  wheel->setHeight(WHEEL_H);
  ball = new LTexture(SCREEN_W,
SCREEN_H);
  ball->loadFromFile("BALL.png");
  ball->setWidth(BALL_W);
  ball->setHeight(BALL_H);
  sound = new Sound;
  FillTheMapsOfRoulette();
}
```

class SpinScreen

```
SpinScreen::~SpinScreen()
{
 roulette->free();
 delete roulette;
 wheel->free();
 delete wheel;
 ball->free();
 delete ball;
}
```

```
bool SpinScreen::Clear()
{
 SDL_RenderClear(LWindow::gRenderer);
 isActive = false; return true;
}
int SpinScreen::GenerateWinningNumber()
{
 srand(time(NULL));
 winningNumber = rand() % 37; return winningNumber;
}
int SpinScreen::GetWinningNumber()
{
 return winningNumber;
}
bool SpinScreen::IsReadyForBonus()
{
 if (numberOfSpins % SPINS_TO_BONUS == 0)
 {
  numberOfSpins = 1; return true;
 } return false;  }
```

```
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];
};
```

```
WinScreen::WinScreen()
{
 background->loadFromFile("WinScreen.jpg");
 background->setWidth(SCREEN_W);
 background->setHeight(SCREEN_H);
 for (int i = 0; i < COIN_COUNT; i++)
 {
  coin[i] = new LTexture(rand() % 200 + 100, rand() %SCREEN_W);
  coin[i]->loadFromFile("coin.png");
  coin[i]->setWidth(COIN_W);
  coin[i]->setHeight(COIN_H);
 }
 fillRectPosition();
}
WinScreen::~WinScreen()
{
 for (int i = 0; i < COIN_COUNT; i++)
 {
  coin[i]->free();
  delete coin[i]; } }
```

class WinScreen

```
bool WinScreen::Draw()
{
 if (background->render(NULL, 0, NULL))
 {
  return true;
 }
 return false;
}
bool WinScreen::Clear()
{
 SDL_RenderClear(LWindow::gRenderer);
 return true;
}
```

```
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);
  } } }
```

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::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); } }
```

ВЪЗТАНОВЯВАНЕ НА ИГРАТА

При натискане на бутон RESUME в началният екран играта се възтановява до момента на прекъсване. Възтановяването се реализира с помоща на XML файлове.

ДЕМО

БЛАГОДАРИМ ЗА ВНИМАНИЕТО