“Спонсорирано обучение по C++”

Финален практически проект

Тема: Reel Slot Game “Space Adventure”

Изготвили проекта:

Симеон Ангелов, Илия Илиев,

Михаел Мушков, Димитринка Вълкова

гр. София

26 Юли, 2017 година

Съдържание Стр

Анализ на изготвеното приложение

Задание: Да се направи Reel Slot игра, с тема “Приключения в космоса”. Всичко по сюжета на играта (фигури, звуци, анимации е по избор на екипа).

Слот игра с 3 реда, 5 колони. Играта има 9 фигури (напълно произволни, по възможност тематични ресурси за фигурите). Ролките се завъртат при натискане на бутон Start/Spin.

Основни състояния: Играта трябва да има пет основни състояния: **Intro** – начален екран с няколко основни бутона за управление и/или настройка). **Game** – основна логика, специфична за конкретната игра. Това състояние може да се характеризира с други подсъстояния, математика, таблици на печалбите, анимации, бутони за управление и/или настройка и др. **Win** – това е състоянието, в което се намира играта при печалба (може да бъде различно за различните типове игри). **Bonus** – представлява състояние на допълнителна печалба, под формата на друга игра и/или мистериозно натрупана сума. **Outro** – в това състояние се влиза по инициатива на играча, след натискане на бутон **CashOut.** По този начин се изплащат текущите пари на играча.

Recovery: Способност за възстановяване – играта трябва да запази всички текущи състояния и настройки в XML файл, за да може при евентуално аварийно събитие (отпадане на захранването или каквото и да е събитие водещо до изключване на приложението) играта да се възстанови там, от където е прекъсната. Използваната библиотека е pugixml.

Intro – всяка игра трябва в това си състояние да има следните бутони:

* *Insert Credit* – с този бутон се добавят кредити в играта (стойността, която ще се добави, може да бъде фиксирана или произволна);

* *Volume* – бутон за регулиране звука на играта;

* *Info* – бутон с кратка информация за правилата на играта и валутата плюс единствената деноминация на която ще се играе. Ако играта има някакви специални фигури или бонуси, то те също трябва да бъдат описани в info страница. Бутонът е активен по всяко време. Всеки екип трябва да избере една от следните деноминации: 0.01, 0.25, 5.00, 2000.00 (пример: ако избера деноминация 0.25 BGN и вкарам 300 кредита от бутона Insert Credit и след това веднага натисна бутона

Cashout, сумата, която ще се изпише на поздравителния екран е 75 BGN).

Start new game – чрез този бутон се нулират всички текущо запазени състояние и/или настройки в XML файла, стартира се нова (чиста) игра и записването започва отново.

Load game – с този бутон се зареждат всички текущо запазени настройки в XML файла, като играта по този начин се възстановява там от където е спряна.

Game – Играта има поле “panel”, който съдържа следните функции:

Bet/Бет поле - играча има възможност да сменя (увеличава или намаля) залога за линия. Максималният залог е 2000 кредита. Необходимо е да има поне 15 стъпки за увеличаване или намаляне на залога като се започне от 1. Налични са бутони + и –;

Lines/Линии поле - играча има възможност да сменя (увеличава или намаля) линиите, на които да се играе. Ако са избрани 5 линии, то всички линии над 5 не печелят нищо.Максималните линии за игра са 15. Налични са бутони + и –;

Max Bet/Максимален Залог - с бутон Max Bet играчът може рязко да смени настройките на бет и линии на техните максимални стойности и с този залог да стартира следващата игра. Полето Total Bet пресмята тоталния залог, на който се играе залог*линии;

Поле *Win* в което се показва единствено спечелената от текущата игра сума. Стойностите в това поле се показват в пари;

Credit - в това поле се показва текущо колко пари има играча;

Бутон *Start/Spin*, който единствено завърта нова игра. Нова игра може да се стартира при напълно завършена стара;

Бутон *Paytable* - показва се отделен екран, съдържащ таблицата с печалбите на фигурите;

Бутон *Cashout* – за изплащане на парите, натрупани в играта;

Позицията на “панел” полето е произволно , като изброените елементи могат да са (по избор) разположени по целия екран.

Win – Вътрешно състояние на играта, което следи каква е печалбата на играча. Налични са 25 линии, по които се определят печалбите от играта.

Всяка фигура има т.нар. “Тежест” или брой фигури, които печелят съответната за фигурата сума. По-голямата печалба на дадена фигура определя нейната по-голяма “тежест”.

Таблицата на печалбите описва печалбата на всяка комбинация от фигури. Минималната комбинация от фигури за печалба е 3, т.е. При попадение на 2 еднакви фигури не се печели нищо. Всички печалби са кредити!!! По време на печалбите анимациите на фигурите не се прекъсват. Прекъсването става, при завъртане на следващата игра.

Bonus – Една от фигурите в играта е “Special Figure”. При 5 специални фигури (по една на всяка ролка), към печалбата се добавя 5 пъти максималният залог. Стартира се поздравителен, Bonus Win Splash като (по възможност) спечелената сума се отброява на този екран. Екранът се скрива/премахва след 6 секунди. Стартира се Bonus игра (double up) като играча има възможност да удвои спечелената сума. Играчът има право на 2 опита за удвояване.

След края на удвояването и преминаване към основна игра се стартира Double Up Win Splash, който показва каква е спечелената сума CAMO, ако удвояването е успешно.

От bonus игра трябва да има възможност да се излиза, ако играчът се откаже да удвоява с бутон Start.

Outro – трябва да се покаже поздравителен екран със спечелената сума, конвертирана от кредити в пари, според избраната деноминация, като този екран трябва да стои 10 секунди и след това да се премине автоматично към екран Intro. Към това състояние се преминава с бутон CashOut от играта.

Кратко описание на избрания подход

По време на етап разработка на текущото приложение, бе избрана трислойната архитектура Model-View-Controller (MVC). Представлява архитектурен шаблон за дизайн (design pattern) в програмирането, основан на разделението между модела на данните, графичния потребителски интерфейс и бизнес логиката в дадено приложение.

Model – Моделът на данните е ядрото на приложението, предопределено от областта, за която се разработва, която в нашия случай е игралната индустрия. Обикновено това са данните от реалния свят, които сме моделирали и над които искаме да извършваме различни операции и манипулации: да въвеждаме, променяме,

визуализираме и т.н. Трябва да отбележим, че съществуват разлики в дефинирането на понятието “свят”. От една страна това е реалният ни обкръжаващ свят, а от друга – въображаемият абстрактен моделен свят, който е продукт на човешкия разум, който възприемаме във вид на твърдения, формули, математическа символика, схеми и други помощни средства.

За това конкретно приложение моделът се състои от кредитите, с които играе играчът, залогът за линия, броят линии и матричното описание на игровите ролки.

View – тази част от приложението е отговорна за показването на данните от модела и представянето им на потребителя. Графичният потребителски интерфейс, ще бъде базиран на open-source библитеката SDL (Simple DirectMedila Layer). Представлява крос-платформен софтуер, предназначена за разработка на игри, управлението на видео, аудио, входни устройства, нишки и зареждането на споделени обекти, мрежови връзки и таймери. Подходяща е за различни приложения и игри. Тук влизат всичките екрани на играта.

Controller – това е междинният слой от архитектурата, който взима данните от модела или извиква допълнителни методи върху него, предварително обработва данните от модела, и чак след това ги предава на изгледа. Тук ще се извърва цялата логика на нашето приложение, включително запазването и възстановяването на данните.

Част от предимствата от използването на MVC са:

- Моделът е независим от контролера и изгледа.
- Моделът може да бъде планиран и осъществен независимо от другите части на системата.
- За един и същи модел могат да бъдат осъществени различни изгледи (интерфейси) – например различни разновидности на играта Reel Slots.
- Контролерът и изгледът могат да бъдат променени, без да се налага промяна в модела.

Функционално описание на приложението

В тази точка ще разгледаме основните функционалности на приложението. Първоначално ще разгледаме модела на данните, след това изгледа и накрая - контролера.

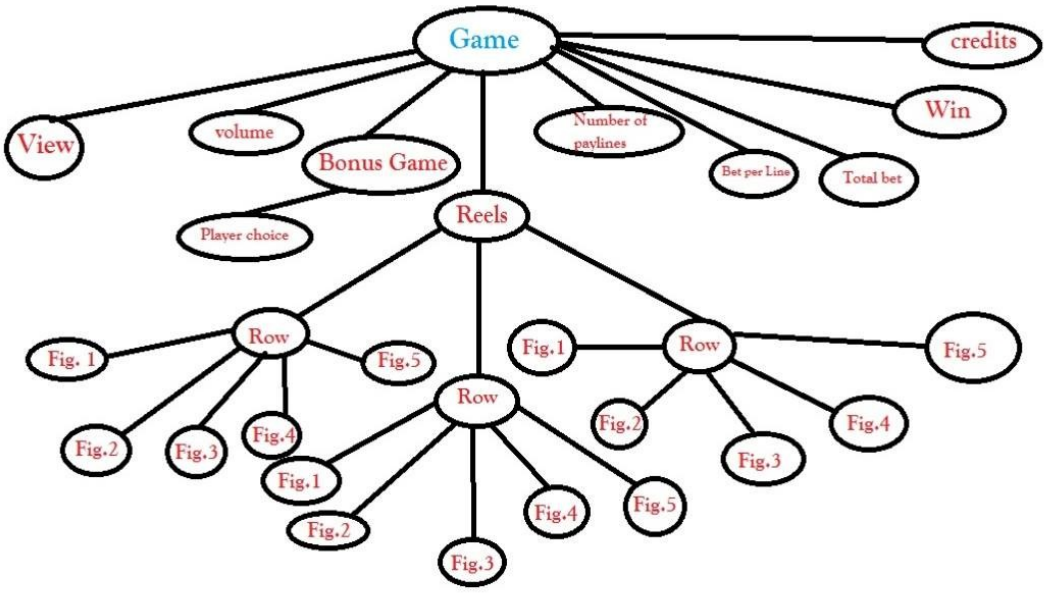
Model

Класът, който държи модела на нашите данни, е GameModel. Съдържа полета от целочислен тип за съхранение на залога за линия, броя линии, печалбата и кредитите. Игровите ролки се съхраняват във вектор от вектори, който служи за описание на редовете и ролките. Векторът е динамична структура, която има променлива дължина и поддържа лесно добавяне откъм края. Лесно може да се предава като аргумент на функция или да се връща. Променливата дължина е подходяща за евентуално добавяне на допълнителни редове или ролки. Самите игрови ролки са от потребителски дефиниран тип – структура, която съдържа масив от пет елемента, които са от изброимия тип Фигура. Предоставени са аксесори и мутатори, които да достъпват полета, както и функция, която да напълни матрицата със стойности по подразбиране.

Recovery

Запазването и възстановяването на данните от XML файл се изпълнява от класа GameRecovery. Данните, които ще се съхраняват във файла, са полета на модела, както и нивото на звука и текущия изглед. Процесите на сериализация и десериализация се изпълняват от външната библиотека pugi, намираща се в папката XML на GameRecovery.

Изпълнени са три основни функционалности- Създаване на празен документ, под формата на дървовидна структура (Document Object Model – DOM Tree). Коренът на дървото е елементът Game. Към него се добавят нови дъщерни елементи – по един за всяко поле, което ще пазим. Интересно е съхранението на матрицата от игрови ролки: към корена се добавя един дъщерен елемент, който е отговорен за цялата матрица. Към него се добавят три елемента – по един за всяка редица, а към всеки от тези три елемента – пет елемента (по един за всяка фигура). Така вътрешно се представя матрицата 3 на 5.



Информацията, която искаме да запазваме в елементите, е под формата на атрибути. Изборът на атрибути, произлиза от възможността за записване на повече от едно поле за елемент (например атрибути Индекс и Стойност).

Създава документ в директорията Save на GameRecovery

- Сериализация - Процесът на запазване на данните от модела в .xml файл. Отделните полета се подават като аргументите на функциите, които актуализират стойностите на атрибутите на съответните елементи на DOM дървото.

- Десериализация – Процесът на четенето на данни от файла, след което се инициализира модела със запазената информация.

View

LifeCycle

Главният клас, управляващ всички изгледи и техните функционалности. Съхранява всички техни бутони, анимации. Отговаря за презентирането на отделните изгледи, както и за музикалните ефекти на цялата игра.

Инизиализира звука и картината, както и обекти от тип TTF_Font, които визуализират информацията от модела на играча.

Създава основния прозорец на играта и неговия SDL Render обект, избирайки шрифта, който ще използваме за текста – Xanadu.

Предвидена е функционалност за зареждане на текстура по избор, може да зарежда необходимите текстури за изглед 1 (началния екран, добавянето на кредитите, бутони за правила и тежест на фигурите); изглед 2 – обща информация за линии и бутон за начало на играта, който отвежда играча към изглед 3 – самата игра, държащ необходимите текстури.

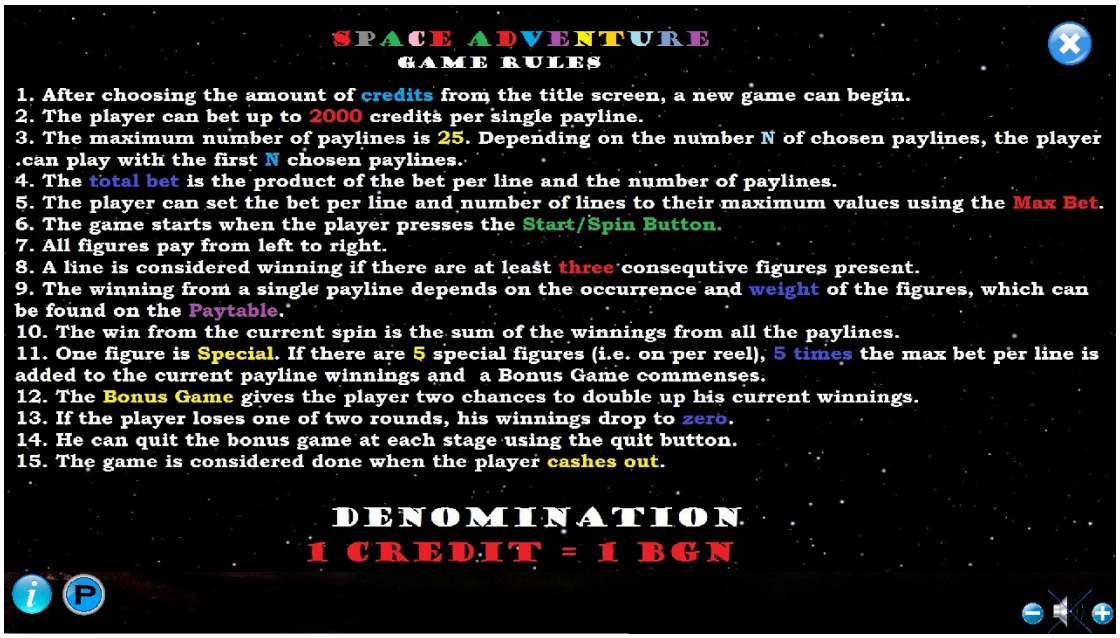
Cashout анимацията се състои от 92 текстури, които се зареждат в самостоятелна функция.

Всички бутони разполагат с музикални ефекти, а самият фон се движи отляво надясно при бутон Напред, а при бутон Назад – отдясно-наляво.

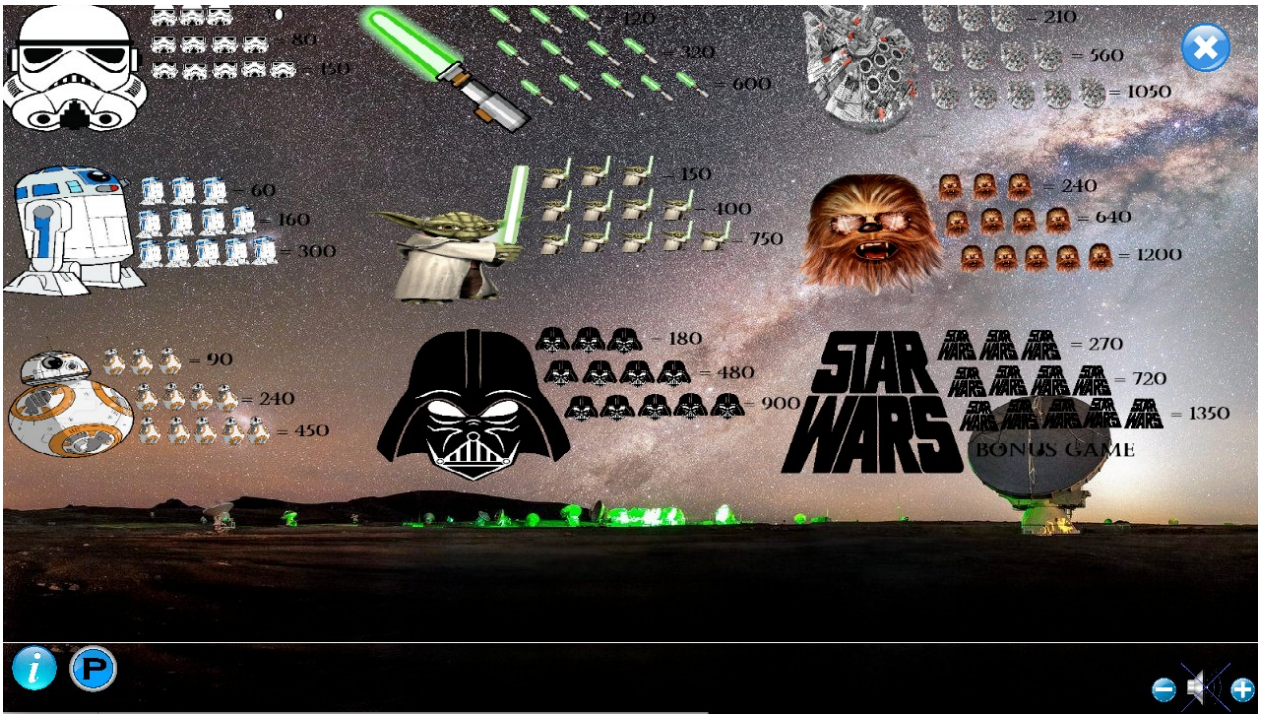
Изглед 1

Съдържа текстури за логото на играта, два бутона за начало на нова игра, продължаване на стара игра, бутон за информация и за визуализация на Paytable, както и бутони за намаляваме/увеличаване на звука. Последните три присъстват на почти всички изгледи, без кредит-менюто.

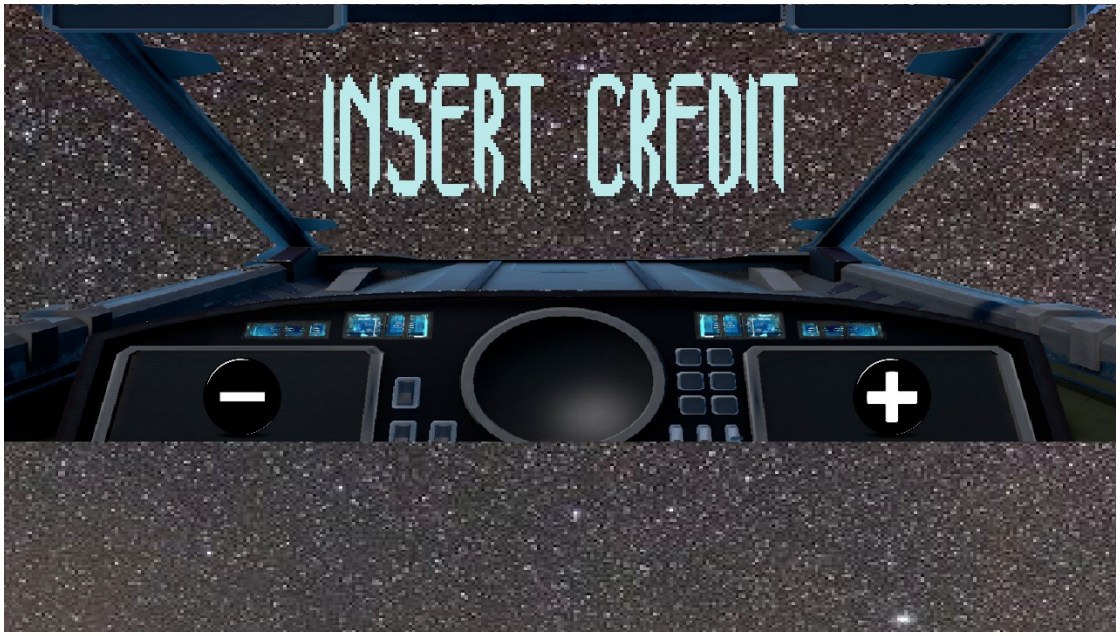
При натискане на бутон информация се спуска падаща текстура отгоре надолу, която показва правилата на играта, заедно с деноминацията. По време на нейната визуализация единствените бутони, които са активни, са тези за звука. Изходът се извършва с бутон “X”.



Бутон Paytable работи аналогично на бутон info. Визуализира информация за печалбите на отделните фигури. Отново са активни само фигурите за управление на звука. Изход се извършва с бутон “X”

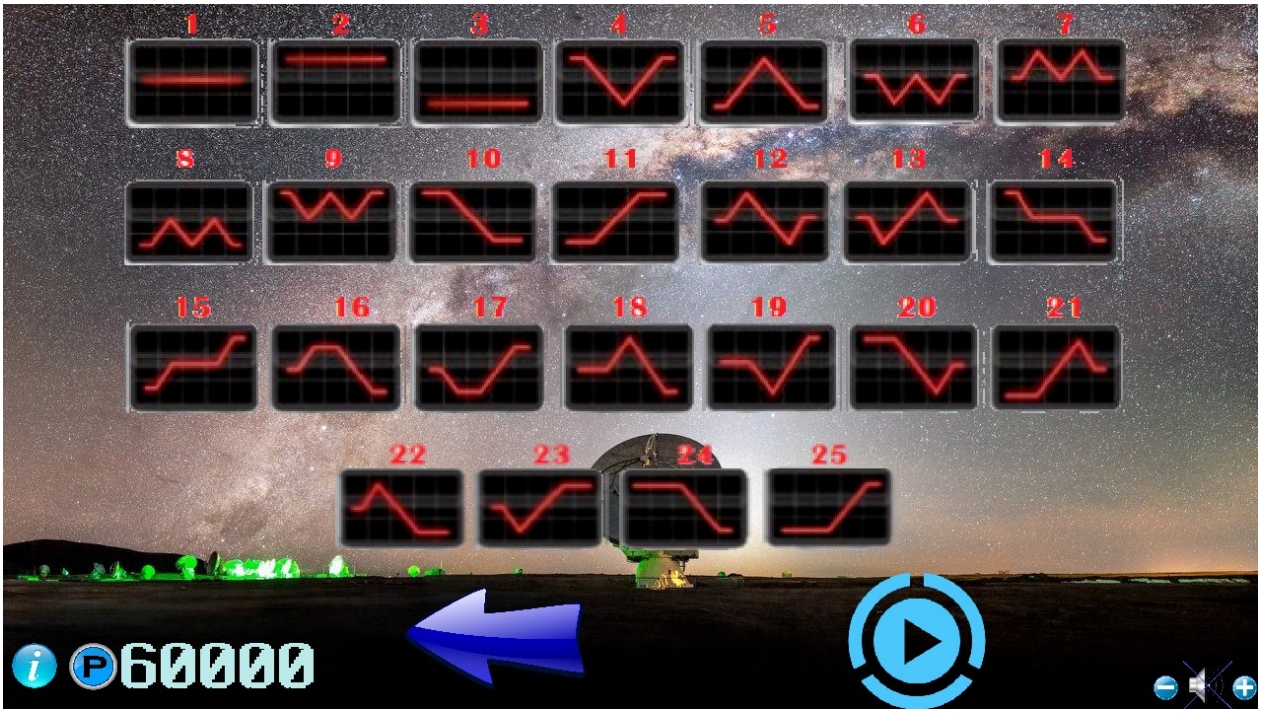


Бутон Start New Game ни отвежда в кредит-менюто. Преходът към него се извършва посредстом приближение(zoom) към горен ляв ъгъл. След това вертикално се спуска текстурата spaceship, която визуализира самото меню. Съдържа бутони за увеличаване и намаляване на кредитите. Самите кредити се визуализират с TTF обект. Заслужава да отбележим, че, при неположителна стойност на кредитите, бутонът OK е неактивен.



Ако е активен, при натискането му, се връщаме на началния екран, на който вече присъства бутонът за изглед 2, който се активирал от булева функция.

Изглед 2



Бутонът Forward ни отвежда към следващия втори изглед. Преходът към него представлява slide анимация отдясно-наляво, която визуализира линиите (разположени на четири реда). Освен тях, играчът може да види с какъв кредит разполага. Има опция да се върне към началния изглед, ако сметне, че сумата му е недостатъчна, или да продължи към изглед 3, където се намира основната игра, посредством бутон Play. Анимацията за изход от екрана се извършва в следната последователност: Първи ред се издига нагоре, скривайки се. Четвърти ред се спуска надолу, като се скрива. Втори и трети се скриват, като съответно отиват наляво и надясно.

Изглед 3

Преходът към изглед 3 – игра: Отново имаме slide на фона. След това от долен десен към горен ляв ъгъл, диагоално се представят текстурата за ротативката и заредените на случаен принцип фигури.

Като приключи анимацията се зареждат и бутоните, които са:

Start/Spin – стартира анимацията за въртене на ролките;

Поле Win визуализира печалбата;

Поле Winning paylines – показва, кои са печелившите линии;

Поле Total Bet – показва общия залог;

Max Bet – бутон, който задава максимални стойности на залога за линия и броя на линии, само ако не надвишава текущите кредити (визуализира и 25 линии на ролките;

Поле Bet с бутони + и -, които увеличават и намаляват залога;

Поле Lines с бутони + и -, които управляват увеличението и намаляването на линиите. Тази промяна се отразява и с визуализация на съответния брой линии върху ролките;

Бутон Cashout, който стартира поздравителна анимация;

Ролки - първата ролка се върти 9 пъти, като тази бройка се увеличва с две за всяка съседна ролка. Така завъртането спира от лявата ролка към дясната.

BonusGame

При пет специални фигури се стартира бонус игра,

Първият екран е Bonus Win Splash, каква е спечелена сума.

След това се зарежда втори екран – бонус игра, която дава възможност на играча да удвои печалбата си. Има два бутона, с които може да избере червена или черна карта. В центъра на екрана са визуализирани две карти, обърнати с гръб. Първата се обръща при натискане на един от двата бутона. Ако играчът не е познал боята на картата, губи печалбата от текущата игра. При познат първи опит има възможност да се откаже или да опита втори път. Аналогично ако познае – печели, ако не – губи.

Третият екран е Double Up Win Splash, която показва спечелената сума, ако удвояването е успешно.

Outro

Анимира 92-те текстури, показва поздрав със спечелените пари, като ги нулира, симулирайки как играчът си прибира печалбата. Анимацията на текстурите се изпълнява в обратен ред, след което се извършва преход към началния изглед.

Controller

IntroController

При този контролер, се създава празният .xml файл. Предвидена е функционалност за добавяне на кредити, чрез увеличаване и намаляване на броя им, след което се запазват във файла.

GameController

Съдържа основната логика на играта. Броят на линиите и залогът за линия могат да се увеличават и намаляват, след което да се завъртят ролките при налични кредити. Използвайки генератор на псевдослучайни числа, се задават произволни стойности на ролките. Изчислява общия залог, както и печалбата. Всички тези промени се отразяват върху модела на данните, използвайки релацията композия

между GameModel и GameController класа. Запазването на информация в XML файл се извършва в ключови моменти като инициализация на матрицата, държаща ролките, или при изчисление на печалбата.

BonusGame

Приемайки модела по адрес като параметър в инициализиращата си функция, този клас може да актуализира печалбата и кредитите му. Използвайки статични методи, не се извиква създаването на обект, за да се използва функционалността му. Тя включва определяне дали опитът за удвояване е успешен и обновяване на печалбата и кредитите, в зависимост от този резултат.

Изпълнение на функционалностите

GameModel

Член-променливи:

- m_matrixGameReels: vector<vector<Figures> >

Вектор, който съдържа в себе си три вектора, отговарящи на редовете на матрицата, като всеки от тях съдържа пет елемента (по един за всяка ролка). Със своята размерност 3x5 тя описва игровите ролки от екрана Game. Тази структура е избрана пред традиционната матрица(масив от масиви), поради по-лесното си предаване на и връщане от функции, както и възможността за добавяне на ролки и редове в бъдеще.

- int m_iNumberOfLines: int

Съдържа броят линии, с които играчът е избрал да играе и залага на тях. Например при 5 избрани линии, играчът залага на линии от първа до пета. Има минимална стойност от 1 и максимална – от 25.

- int m_iBetPerLine: int

Съдържа залога за единична линия. В границите от 1 до 2000.

- int m_iWin: int

Държи печалбата от линиите, с които е избрал да играе потребителят за текущата игра, както и евентуално печалба от бонус игра. При стартиране на нова игра се инициализира с нула.

- int m_iCredits: int

Основната валута на играта са кредити. Инициализират се в началния екран, а в основната и бонус игра от тях се изважда общият залог и се добавя печалбата.

- int m_iTotalBet: int

Общият залог за една игра е произведението от залогът за единична линия и броят линии, избрани от играча

- vector<Payline> m_vecPaylines: int

Вектор, който да държи 25 линии - инстанции на структурата Payline. Тя е потребителски тип, съдържащ като поле масив от изброим тип (enum) Figure с 5 елемента – по един за всяка фигура (описана с енумератор). Статична е, за бъде инициализирана само веднъж за цялата игра.

Член-функции:

* Аксесорите и мутаторите на член-променливите ще бъдат пропуснати.

+ GameModel(iNumberOfLines = 1: int, iBetPerLine = 1: int, iWin = 0: int, iCredits = 0: int, iTotalBet = 1: int)

Дефинирана на 13 ред. Конструктор по подразбиране на класа, който инициализира със стойности по подразбиране полетата на класа.

+ InitDefaultReels(): void

Дефинирана на 23 ред. Първо създаваме локален вектор, който отговаря за ред от матрицата. Външният цикъл итерира 3 пъти – по един за всеки ред. Вътрешният цикъл прави 5 итерации, като на всеки пас една променлива от тип Figure се инициализира със стойност по подразбиране и се поставя откъм края на вектора. След приключване на вътрешния цикъл vecCurrentRow се поставя откъм края на m_matrixGameReels. Така матрицата се инициализира със стойности по подразбиране.

+ SetReelElement(figure: const Figures&, iRow: int, iCol: int) : void

Дефинирана на ред 72. Тази функция променя единичен елемент от матрицата. Приема три параметъра – фигурата, която ще запишем, редът и колоната от матрицата, където ще я запишем. Вътрешни проверки следят дали последните два аргумента са в границите на матрицата.

+ GetReelElement(iRow: int, iCol: int) : Figures

Дефинирана на ред 53. Връща елемент от матрицата по подадени ред и колона. Вътрешни проверки следят дали последните два аргумента са в границите на матрицата.

LifeCycle

Член – променливи:

+Rect : RectStruct

// Rects View 1 ->

+ rectBackground : SDL_Rect - създаваме ект, на който ще прикрепим бекграунд текстурата

+ rectForwardButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за напред

+ rectLogo : SDL_Rect - създаваме ект на който слагаме текстурата на логото на играта ни

+ rectInsertCredit : SDL_Rect - създаваме ект, на който прикрепяме текстурата за бутона за добавяне на кредит

+ rectInfoButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за информация

+ rectVolume : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона на звук

+ rectVolumePlusButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за увеличаване звука

+ rectVolumeMinusButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за намаляване на звук

+ rectSpaceShip : SDL_Rect - създаваме ект, на който прикрепваме текстурата за кредит менюто

+ rectSpaceShipButtonMinus : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за намаляване на кредит

+ rectSpaceShipButtonPlus : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за добавяне на кредит

+ rectSpaceShipButtonOk : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за потвърждаване на кредитите

+ rectTextCredit : SDL_Rect - създаваме ект, на който ще стои ТТФ формата за визуализация на кредит

+ rectRules : SDL_Rect - създаваме ект, на който прикрепваме текстурата за правилата на играта

+ rectCloseRulesButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за затваряне на екрана с правилата

// Rects View 2 ->

+ rectBackButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата за бутона за връщане назад

+ rectTextCreditController2 : SDL_Rect - създаваме ект, на който слагаме ТТФ формата за кредит визуализацията на второ ВЮ

+ rectFirstLine : SDL_Rect - създаваме ект, на който прикрепваме текстурата за линиите за печалби (първи ред)

+ rectSecondLine : SDL_Rect - създаваме ект, на който прикрепваме текстурата за линиите за печалби (втори ред)

+ rectThirdLine : SDL_Rect - създаваме ект, на който прикрепваме текстурата за линиите за печалби (трети ред)

+ rectFourthLine : SDL_Rect - създаваме ект, на който прикрепваме текстурата за линиите за печалби (четвърти ред)

+ rectButtonPlay : SDL_Rect - създаваме ект, на който прикрепваме текстурата на бутона ПЛЕЙ, който ни отвежда към третото основно меню на играта

// Rects View 3 ->

+rectSlot : SDL_Rect - създаваме ект, на който прикрепваме текстурата за слот машината

+ rectMinusButtonLines : SDL_Rect - създаваме ект, на който прикрепваме текстурата на бутона за намаляване на линиите за игра

+ rectPlusButtonLines : SDL_Rect - създаваме ект, на който прикрепваме текстурата на бутона за увеличаване на линиите за игра

+ rectMinusBetButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата на бутона за намаляване залога

+ rectPlusBetButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата на бутона за увеличаване на залога
+ rectMaxBetButton : SDL_Rect

+ rectStartSpinButton : SDL_Rect - - създаваме ект, на който прикрепваме текстурата на бутона Старт/Спин

+ rectCashOutButton : SDL_Rect - създаваме ект, на който прикрепваме текстурата на бутона за вземане на текущата печалба

+ rectWhiteScreen : SDL_Rect - създаваме ект, на който прикрепваме текстурата за фон на КЕШОУТ анимацията

+ rectFinalCreditText : SDL_Rect - създаваме ект, на който прикрепваме ТТФ формата, за визуализация на печалбата

// Rectangles for slot figures

// Rectangles for FIRST row

+ rectFigure1Slot1 : SDL_Rect - създаваме ект, на който прикрепваме текстурата на първата игрална фигура на първия слот

+ rectFigure2Slot1 : SDL_Rect - създаваме ект, на който прикрепваме текстурата на втората игрална фигура на първия слот

+ rectFigure3Slot1 : SDL_Rect - създаваме ект, на който прикрепваме текстурата на третата игрална фигура на първия слот

//Rectangles for SECOND row

+ rectFigure1Slot2 : SDL_Rect - създаваме ект, на който прикрепваме текстурата на първата игрална фигура на втория слот

+ rectFigure2Slot2 : SDL_Rect - създаваме ект, на който прикрепваме текстурата на втората игрална фигура на втория слот

+ rectFigure3Slot2 : SDL_Rect - създаваме ект, на който прикрепваме текстурата на третата игрална фигура на втория слот

// Rectangles for THIRD row

+ rectFigure1Slot3 : SDL_Rect - създаваме ректъ, на който прикрепваме текстурата на първата игрална фигура на третия слот

+ rectFigure2Slot3 : SDL_Rect - създаваме ректъ, на който прикрепваме текстурата на втората игрална фигура на третия слот

+ rectFigure3Slot3 : SDL_Rect - създаваме ректъ, на който прикрепваме текстурата на третата игрална фигура на третия слот

// Rectangles for FOURTH row

+ rectFigure1Slot4 : SDL_Rect - създаваме ректъ, на който прикрепваме текстурата на първата игрална фигура на четвъртия слот

+ rectFigure2Slot4 : SDL_Rect - - създаваме ректъ, на който прикрепваме текстурата на втората игрална фигура на четвъртия слот

+ rectFigure3Slot4 : SDL_Rect - създаваме ректъ, на който прикрепваме текстурата на третата игрална фигура на четвъртия слот

//Rectangles for FIFTH row

+ rectFigure1Slot5 : SDL_Rect - създаваме ректъ, на който прикрепваме текстурата на първата игрална фигура на петият слот

+ rectFigure2Slot5 : SDL_Rect - създаваме ректъ, на който прикрепваме текстурата на втората игрална фигура на петият слот

+ rectFigure3Slot5 : SDL_Rect - създаваме ректъ, на който прикрепваме текстурата на третата игрална фигура на петият слот

// Info Panel

+ rectLinesText : SDL_Rect - създаваме ректъ, на който прикрепваме TTФ формата за визуализацията на линиите

+rectLinesShow : SDL_Rect - създаваме ректъ, на който прикрепваме TTФ формата за визуализация броя с колко линии играем

+ rectBetText : SDL_Rect - създаваме ректъ, на който прикрепваме TTФ формата за визуализацията на залога на играта

+ rectBetLabel : SDL_Rect - създаваме ект, на който прикрепваме ТТФ формата, който посочва БЕТ полето за игра

+ rectTotalBetText : SDL_Rect - създаваме ект, на който прикрепваме ТТФ формата за визуализацията на линиите

+ rectGoodLuckText : SDL_Rect - създаваме ект, на който прикрепваме ТТФ формата за визуализация на печелившите линии

+ rectWinText : SDL_Rect - - създаваме ект, на който прикрепваме ТТФ формата за визуализацията на печалбата

- m_nWidth : int const - поле, което не го променяме, защото съдържа ширината на екрана

- m_nHeight : int cosnt - поле, което не го променяме, защото съдържа височината на екрана

- windowPtr : SDL_Window* - създаване на основният прозорец на играта

- rendererPtr : SDL_Renderer* - създаване на основната преработка на играта

- Xanadu : TTF_Font* - създаване на основният фонт за ТТФ на играта

- color : SDL_Color - създаване на цвят, който ще ползваме за ТТФ фонта на играта

- black : SDL_Color - създаване на черен цвят, който ползваме за ТТФ фонта на играта

- creditSurface : SDL_Surface* - създаване на платформа, която ни служи за преход на кредит визуализацията към текстура

- creditTexture : SDL_Texture* - създаване на текстура за кредти визуализацията

- finalCredit : SDL_Surface* - платформа за визуализация на кредита на Аутро екрана

- finalCreditTexture : SDL_Texture* - създаване на текстура за кредита на Аутро екрана

- introMusic : Mix_Chunk* - проектиране на началната музика на играта

- buttonSound : Mix_Chunk* - проектиране на музикален ефект за бутоните

- nextViewSound : Mix_Chunk* - създаване на музикален ефект при преминаване към следващ изгледн на играта

- linesSound : Mix_Chunk* - създаване на музикален ефект за игралните линии

// Information Table ->
// Lines Info

- linesLabelText : string - създаване на поле, което визуализира местоположението на игралните линии

- linesText : string - текст, който показва броя на избраните от потребителя игрални линии

- linesLabelSurface : SDL_Surface* - създаване на платформа за преход на игралните линии към текстура

- linesLabelTexture : SDL_Texture* - създаване на текстура за игралните линии

- linesSurface : SDL_Surface* - създаване на платформа за визуализация на линиите и прехода й към текстура

- linesTextureText : SDL_Texture* - създаване на текстура за за визуализация на линиите

// Bet Info

- betLabel : string - създаване на информационно меню, което ни известява за местоположението на БЕТ менюто

- betText : string - създаване на информационно меню, което ни известява за залога на линия с който играем

- betLabelSurface : SDL_Surface* - създаване на платформа за ТТФ, за преход към текстура

- betLabelTexture : SDL_Texture* - създаване на текстура, която ни известява за залога за линия с който играем

- betSurface : SDL_Surface* - създаване на БЕТ платформа, с която да преминем към текстура

- betTextureText : SDL_Texture* - създаване на БЕТ текстура

// Good Luck Panel – панел съдържащ информация за печелившите линии

- goodLuck : string - създаване на поле, което известява играча за броя спечелени линии на СПИН

- goodLuckSurface : SDL_Surface* - създаване на платформа, която фонта ТТФ изисква, за преминаване към текстура

- goodLuckTextureText : SDL_Texture* - създаване на текстура за информация предоставяща на играча, броя на спечелените линии на врътка, както и техният номер

// Total Bet Info

- totalBetCalc : int - изчисляване общият залог (броя на линиите,които играча е посочил умножен по залога за линия, който е избрал

- totalBetText : string - създаваме поле, което посочва местоположението на екрана, на който визуализираме броя спечелени линии, както и техният номер

- totalBetSurface : SDL_Surface* - създаване на платформа, която изисква ТТФ фонта, за да можем да направим преход към текстура

- totalBetTextureText : SDL_Texture* - създаване на текстура, която ще показва на потребителя общият залог

// Win Panel

- win : string - визуализира на играча, печалбата

- winSurface : SDL_Surface* - създаване на платформа, която изисква ТТФ за преминаване към текстура, която да показва печалбата на на играча
- winTextureText : SDL_Texture* - създаване на текстура, която да показва печалбата на играча

- m_nCredit : int - променлива, която ни служи за работа с кредити

- m_nLinesCounter : int - брояч, който ни преброява броя на линиите за игра

- m_vecSlotFigures : vector<SDL_Texture*> - вектор, съдържащ текстурите на всяка фигура

- m_vecLines : vector<SDL_Texture*> - вектор, съдържащ текстури на всички игрални линии

- vecAnimationCashOut : vector<SDL_Texture*> - вектор, който държи текстурите за КЕШ ОУТ анимацията

// Creating Objects

- Intro : Intro - създаване на обект от Интро ВЮ

- PayLines : Paylines - създаване обект от Второ ВЮ

- Slot : Slot - създаване на обект от трето вю

// Logic Object

- introController : IntroController - създаване на обект от Интрокотролер

- gameController : GameController - създаване на обект от Геймконтролер

- gameRecovery : GameRecovery - създаване на обект от Геймрекъвъри

// Logic Variables ->
- counterViewControllers : int - брояч, който следи на кое ВЮ е потребителя

- quitCycle : bool - премахване на ненужните текстури, прозорци и т.н.

// Bool SpaceShip is Hidden

- spaceShipHidden : bool - следи дали кредит менюто се показва

- StartCreditMenuEffect : bool - булева, която не позволява функционалността на ненужните бутони
- presentSpaceShip : bool - променлива, която следи дали е визуализирано кредит менюто

- okPushedSpaceShip : bool - променлива, която проверява, натиснат ли е бутона OK на кредит менюто

- creditInserted : bool - променлива, която следи имаме ли добавен кредит

- counterVolume : int - брояч, който следи силата на звука

- infoShown : bool - променлива, която следи за инфо бутона, дали присъства в кредит менюто

- startPressed : bool - проверка дали бутона СТАРТ е натиснат

- creditText : string - променлива, която отговаря за визуализацията на кредита

- xZ : int - променлива следяща координатите на мишката по Х

- yZ : int - променлива следяща координатите на мишката по У

- row1fig1 : int - row5fig3 : int - променливи, които държат индекса на случайно избраните фигури на слота

- ev - SDL_Event - създаване на променлива, която следи за събитията, които се случват

+ LifeCycle() – constructor - конструктур

+ InitMedia : void - инициализира и подготвя работата на СДЛ, с компилатора

+ Play() : void - основната функция, която съдържа жизнения цикъл на играта

+ QuitGame() : void - функция, която премахва ненужните текстури, прозорци и т.н

+ IsInsertCreditPressed(x: int, y: int): bool - променлива, която следи, имаме ли вкаран кредит, когато натиснем на определени координати

+ IsInfoPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за информация, когато натиснем на определени координати

+ IsVolumePlusPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за увеличаване на звук, когато го натиснем на определени координати

+ IsVolumeMinusPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за намаляване на звук, когато го натиснем на определени координати

+ IsForwardPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за предвижване напред, когато го натиснем на определени координати

+ IsShipVolumeMinusPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за намаляване на звук, когато го натиснем на определени координати

+ IsShipOkPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за потвърждение, когато го натиснем на определени координати

+ IsShipVolumePlusPressed(x: int, y: int): bool- променлива, която следи, натиснат ли е бутона за увеличаване на звук, когато го натиснем на определени координати.

// View 2

+ IsBackPressed(x: int, y: int): bool- променлива, която следи, натиснат ли е бутона за връщане на старт изгледа ни, когато го натиснем на определени координати

+ IsPlayPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за старт на играта, когато го натиснем на определени координати

// View 3

+ IsStartSpinPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона СТАРТ/СПИН, когато го натиснем на определени координати

+ IsMinusLinesPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за намаляване броя на линии за игра, когато го натиснем на определени координати

+ IsPlusLinesPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за увеличаване на линиите за игра, когато го натиснем на определени координати

+ IsMinusBetPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за намаляване линиите за игра, когато го натиснем на определени координати

+ IsPlusBetPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за увеличаване на залога, когато го натиснем на определени координати

+ IsMaxBetPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е бутона за максимален залог, когато го натиснем на определени координати

+ IsCashOutPressed(x: int, y: int): bool - променлива, която следи, натиснат ли е КЕШОУТ бутона, когато го натиснем на определени координати

// Utility Methods ->

+ LoadTexture(path: string): SDL_Texture* - метод за зареждане на текстури

+ operator = (newObject: SDL_Rect) : SDL_Rect*
+ Object : SDL_Rect* - овърлоад оператор = , която приема обект от СДЛ_РЕКТ и връща нов такъв

+ createRect(x: int, y: int, w: int, h: int) : SDL_Rect - функция, която създава ректа. На посочените от нас координати и с определени от нас размери

+ PreparingTexturesView1():void - зарежда необходимите текстури само за първото ВЮ

+ PreparingTexturesView2():void - зареждане на текстурите сам за второто ВЮ

+ PreparingTexturesView3():void - зареждане на текстурите само за третото ВЮ

+ PreparingTexturesCashOutAnimation():void - зареждане на текстурите само за кеш аут анимацията

+ PreparingTextures():void - зареждане на всички необходими на играта текстури

+ CreateRectsInStruct():void- създава всички необходими ректове, които използваме в играта

+ LoadMusic():void - зареждане на музикалните ефекти

+ PresentView1():void - метод, който добавя всички необходими текстури и функционалности за Изглед1

+ CreditMenu(spaceShipHidden: bool): void - метод, който за всички необходими, текстури и функционалности на кредит менюто

+ PresentCreditMenu(presentSpaceShip: bool): void – метод, който презентира кредит менюто

+ CreditMenuZoomOut(okPushed: bool): void – метод, отговарящ за анимацията на кредит менюто (приближение чрез while цикъл, като увеличаваме width and height на background(фон) текстурата)

+ PassingToView2():void - метод, който подготвя всички текстури и необходими функционалности, нужни за Изглед2

+ PresentView2():void - отвежда потребителя къмИзглед2
+ LinesEffectComing():void - метод, който анимира четирите текстури, съдържащи линиите за игра, използващи четири while цикъла и промяна на координатите на всяка текстура

+ BackToView1():void - метод, който позволя връщане към Изглед1

+ PassingToView3():void - метод, след който линиите се прибират, чрез анимация и минаваме към основното трети изглед на играта

+ PresentView3():void - метод, който презентира трети изглед

+ SpinningReels():void - функция, която отговаря за анимирането на въртенето на всеки слот със 9-те текстури на всяка фигура, като използваме главно while цикъл и броячи, като на първата ролка се спира на 9-тото завъртане, а всяка следваща със стъпка +2 (11,13,15,17)

+ CashOutAnimation():void - метод съдържащ, 92 текстури, които чрез while цикъл се анимират и изписват печалбата на играча

+ WhiteScreenAndCashOutAnimation():void - метод, съдържащ два while цикъла, които „ въртят „ по координати 92-те текстури от кешоут анимацията

View 1 Intro
Член – променливи:

-backgroundTexture : SDL_Texture – зареждане на backround текстурата;

-logoTexture : SDL_Texture – зареждане на Лого текстура;

-insertCreditTexture : SDL_Texture – зареждане на текстура за Инсерт Кредит

-insertCreditPushedTexture : SDL_Texture – зареждане на текстура, която да анимира Инсерт кредит текстурата;

-infoTexture : SDL_Texture – зареждане на ИнфоБутон текстурата;

- infoTexturePushed: SDL_Texture – зареждане на текстура, която анимира ИНФО бутона текстурата;

-volumeTexture : SDL_Texture – зареждане текстура на звук;

-volumeTexture1 : SDL_Texture - зареждане текстура на звук;

-volumeTexture2 : SDL_Texture - зареждане текстура на звук;

-volumeTexture3 : SDL_Texture - зареждане текстура на звук;

-volumePlusButton : SDL_Texture – зареждане текстура за увеличаване на звук;

-volumePlusButtonPushed : SDL_Texture – зареждане текстура, която анимира ПЛЮС бутона за увеличаване на музиката;

-volumeMinusButton : SDL_Texture – зареждане текстура за намаляване на звук;

-volumeMinusButtonPushed : SDL_Texture – зареждане тесктура, която анимира МИНУС бутона на звука;

-forwardButtonTexture : SDL_Texture – зареждане текстура на бутон за преминаване в следващото ВЮ(VIEW);

-pushedForwardButtonTexture : SDL_Texture – зареждане на текстура, която анимира натискане на бутона за преминаване в следващо ВЮ;

-spaceShipTexture : SDL_Texture – зареждане текстура за кредит меню;

-spaceShipButtonPlus : SDL_Texture – зареждане текстура за плюс бутон на кредит меню;

-spaceShipButtonPlusPushed : SDL_Texture – анимиране на бутона за ПЛЮС на кредит меню;

-spaceShipButtonMinus : SDL_Texture – зареждане текстура за бутон за намаляване на кредит;

-spaceShipButtonMinusPushed : SDL_Texture – анимира бутона за намаляване на кредита в кредит менюто;

-spaceShipButtonOK : SDL_Texture – зарежда текстура за бутон ОК на кредит менюто;

-spaceShipButtonOkPushed : SDL_Texture – анимира бутона ОК в кредит менюто;

-rulesTexture : SDL_Texture – зарежда текстура за правилата на играта;

-closeRulesButton : SDL_Texture – текстура за бутона за затваряне на екрана с правилата;

View 2 _ PayLines

Член – променливи:

-backButtonTexture : SDL_Texture – зареждане на текстура за БЕК бутона;

-backButtonTexturePushed : SDL_Texture – зареждане на текстура за анимиране на БЕК БУТОН за текстура;

-playButtonTexture : SDL_Texture – зареждане текстура за ПЛЕЙ;

-playButtonTexturePushed : SDL_Texture – зареждане на текстура която анимира бутона ПЛЕЙ;

-firstRow : SDL_Texture – зареждане на текстура за първия ред линии;

-secondRow : SDL_Texture - зареждане на текстура за втория ред линии;

-thirdRow : SDL_Texture - зареждане на текстура за третия ред линии;

-fourthRow : SDL_Texture - зареждане на текстура за четвърти ред линии;

VIEW 3 – Slot

Член – променливи:

-slotTexture : SDL_Texture; - зареждане на текстурата за СЛОТ;

-startSpinButton : SDL_Texture; - зареждане текстурата на СТАРТ СПИН бутона;

-startSpinButtonPushed : SDL_Texture; - зарежда текстура, за анимиране на СТАРТ/СПИН бутона;

-maxBetButton : SDL_Texture; - зареждане текстура за МАКС бет бутона;

-maxBetButtonPushed : SDL_Texture; - зареждане на текстура за анимация на МАКС БЕТ бутона;

-plusLinesButton : SDL_Texture; - зареждане текстура за ПЛЮС бутона за увеличаване на линиите;

-plusLinesButtonPushed : SDL_Texture; - зарежда текстура, която анимира натиснат бутон за добавяне на линии;

-minusButtonLines : SDL_Texture; - зареждане текстура за бутон за намаляване на линии;

-minusButtonLinesPushed : SDL_Texture; - зарежда текстура за анимацията на натиснат бутон за намаляване на линии;

-minusButtonBet : SDL_Texture; - зареждане текстура за бутон за намаляване на залога;

-minusButtonBetPushed : SDL_Texture; - зарежда текстура за анимация на бутона за намаляване на залога;

-plusButtonBet : SDL_Texture; - зарежда текстура за бутона за увеличаване на залога;

-plusButtonBetPushed : SDL_Texture; - зарежда текстура за анимация на бутона за увеличаване на залога;

-cashOutButton : SDL_Texture; - зарежда текстура за КЕШ оут бутон;

-cashOutButtonPushed : SDL_Texture; - зарежда текстура за анимация, че е натиснат кеш оут бутона;

-whiteScreenTexture : SDL_Texture; - зареждане текстура (бяла), която служи за фон на анимацията за кеш оут;

-eFigure1 : SDL_Texture – eFigure9 : SDL_Texture; - зареждане на текстури за фигурите на слота;

-line1 : SDL_Texture – line25 : SDL_Texture; - зарежда текстури за линиите;

Controller

IntroController

Член – променливи:

- m_iCredits: int – променлива, която запазва броя на кредитите на играча;

Член – функции:

+ IntroController(); - конструктор;

+ virtual ~IntroController() - деструктор

+PlayNewGame(): void – метод, който създава празен XML файл;

+IncreaseCredits(): void – метод, който увеличава кредитите със 10 000 всеки път, когато е извикан;

+DecreaseCredits(): void – метод, който намалява кредитите с 10 000, всеки път когато е извикан;

+SaveCredits(): void – метод, запазващ кредитите в XML файл;

GameController

Полета:

- m_baseGame: GameModel

Използвайки шаблона Модел-Изглед-Контролер, създаваме обект от модела, върху който контролерът ще извършва различни операции.

- m_iBetStep: int

Поле, което служи за индексиране на вектора от залози за линия.

- m_vecBetPerStep: vector<BET>

Този вектор съдържа 16 елемента, отговарящи на 16 стъпки, по които се увеличава и намалява залогът, като на всяка стъпка отговаря определена сума. Типът на вектора е изброимия тип BET. Сумите на стъпките са както следва: 1, 2, 5, 10, 25, 50, 75, 100, 250, 500, 750, 1000, 1250, 1500, 1750 и 2000 кредита.

- m_iBonusCounter: int

Полето случи за следене на броя специални фигури. Ако на ролка се падне специална фигура, броячът се инкрементира. Използва се като условие за проверка дали да се стартира бонус играта.

- m_vecWinningPaylines: vector<int>

Векторът държи номерата на линиите, чиято печалба е положителна

Член-функции:

+ GameController()

Дефинирана на ред 23. Конструктор по подразбиране, който извиква конструктора по подразбиране на GameModel, инициализирайки полетата му.

+ virtual ~GameController():void

Дефинирана на ред 28. Деструктор

+ NewGame():void

Дефинирана на ред 34. Ако съществува XML файл, кредитите от него се взимат, посредством функцията GameRecovery::LoadCredits() и се присвояват на полето m_iCredits на модела чрез неговия set метод.

+ LoadGame():void

Дефинирана на ред 46. Зарежда полетата на модела с информацията от XML файла, с помощта на GameRecovery::LoadGameModel() метода, който приема като параметър m_baseGame по адрес. След това InitPaylines инициализира вектора, съдържащ игровите линии, с фигурите от игровите ролки.

- SetDefault(): void

Дефинирана на ред 58. Присвоява нула на броячът m_iBonusCounter и на m_iWin полето на модела, посредством GameModel::SetIWin(). Вика SetTotalBet(), за да актуализира общия залог.

+ Spin():void

Дефинирана на ред 64. Тялото се изпълнява, само ако общият залог е по-малък или равен на кредитите. Извиква методите:

- AddTotalBetToCredits(), който изважда общия залог от кредитите;
- InitCurrentReels(), който инициализира игровите ролки;
- InitCurrentPaylines(), който инициализира игровите линии
- SetTotalWin(), който изчислява текущата печалба и стартира бонус игра, ако е изпълнено условието от пет специални фигури и
- GameRecovery::UpdateGameModel(), който приема като параметър модела по адрес и го запазва в XML файл.

- InitCurrentReels():void

Дефинирана на ред 83. Извиква InitRandomReels(), за да инициализира игровите ролки с произволни фигури от 0 до 7. След това се вика SetSpeciaFigure, която на случаен принцип слага специалната фигура 8 на ролка. За тестови цели може да се извика и StartBonusGame, която да сложи специалната фигура на всяка ролка, за да се стартира бонус играта.

- InitRandomReels():void

Дефинирана на ред 94. С помощта на два вложени цикъла се обхождат игровите ролки. В тялото на вътрешния цикъл се избира случайна фигура между 0 и 7 с помощта на библиотечната функция rand() и заедно с текущия ред и колона се подават на GameModel::SetReelElement(), който актуализира текущата фигура.

- SetSpecialFigure():void

Дефинирана на ред 114. С помощта на два вложени цикъла се обхождат игровите ролки. В тялото на вътрешния цикъл се избира случайна фигура между 0 и 8 с помощта на библиотечната функция rand(). Ако тази фигура е 8 или специална, се присвоява на текущия елемент с помощта на GameModel::SetReelElement(). След това се инкрементира m_iBonusCounter и вътрешния цикъл се прекъсва, защото трябва да има само по една специална фигура на ролка. Така на всяка ролка произволно се присвоява специална фигура.

- StartBonusGame():void

Дефинирана на ред 139. Тестова функция. С помощта на два вложени цикъла се обхождат игровите ролки. В тялото на вътрешния цикъл се избира случаен ред между 0 и 2 с помощта на библиотечната функция rand() и заедно със специалната (последната осма) фигура и текущата колона се подават на GameModel::SetReelElement(). След това се инкрементира m_iBonusCounter. Така на всяка ролка се присвоява специална фигура на произволен ред, за да се старира бонус играта.

- SetTheSameFigures():void

Дефинирана на ред 155. Тестова функция. Използвайки библиотечната функция rand(), се избира една случайна фигура между 0 и 8. С помощта на два вложени цикъла се обхождат игровите ролки и на текущия елемент се присвоява случайната фигура с помощта на GameModel::SetReelElement().

- InitCurrentPaylines():void

Дефинирана на ред 173. Извиква се методът ErasePaylines(), след което инициализира 25-те линии с елементи от игровите ролки.

+ MaxBet():void

Дефинирана на ред 206. Ако кредитите са по-големи или равни от максималния залог, функцията присвоява максимални стойности на залога за линия и броя на линиите.

+ IncreasePaylines():void

Дефинирана на ред 221. Ако броят на линиите не надвишава максималната стойност и не надвишава кредитите, се подава като параметър на GameController::SetNumberOfPaylines(), който вика set метода на модела и обновява полето m_iNumberOfLines.

+ DecreasePaylines():void

Дефинирана на ред 243. Ако броят на линиите е положителен, се намалява с единица. Извиква се SetTotalBet(), за да се актуализира общия залог.

- PaylinesExceedCredits(iStep: int): bool

Дефинирана на ред 264. Параметърът е броят на линиите, увеличен с единица. Използва се за изчисление на новия общ залог. Връщаният резултат е дали общият залог надвишава кредитите.

+ IncreaseBet():void

Дефинирана на ред 279. Ако залогът за линия не надвишава 2000, полето m_iBetStep се инкрементира. Подава се като параметър на функцията BetExceedsCredits. Ако върне false, се подава като параметър на GameController::SetBetPerPayline() метода, който задава като стойност на полето m_iBetPerLine на модела, сумата от вектора m_vecBetPerStep, отговаряща на подадения аргумент. Извиква се SetTotalBet(), за да се актуализира общия залог.

+ DecreaseBet():void

Дефинирана на ред 297. Ако залогът за линия е по-голям от нула, полето m_iBetStep се декрементира. Подава се като параметър на GameController::SetBetPerPayline() метода, който задава като стойност на полето m_iBetPerLine на модела, сумата от вектора m_vecBetPerStep, отговаряща на подадения аргумент. Извиква се SetTotalBet(), за да се актуализира общия залог.

- BetExceedsCredits(iStep: int): bool

Дефинирана на ред 321. Параметърът е стъпката, по която ще се изчисли новият залог за линия. Използвайки новия залог за линия, за да се изчисли новия общ залог. Връща като резултат дали новият общ залог надвишава текущите кредити.

- SetTotalBet():void

Дефинирана на ред 336. Вика set метода на модела, който присвоява на полето m_iTotalBet продукта от броя линии и залогът за линия. Когато се извика този метод, общият залог се актуализира.

- AddTotalBetToCredits():void

Дефинирана на ред 342. Ако текущият залог не надвишава кредитите, разликата между текущите кредити и общия залог се присвоява на полето m_iCredits на модела посредством неговия set метод.

+ TotalBetExceedsCredits(): bool

Дефинирана на ред 356. Проверява дали общият залог е по-малък от нула или надвишава текущата стойност на кредитите.

- SetTotalWin():void

Дефинирана на ред 364. Извиква методите WinFromPaylines и AddWinToCredits, които изчисляват печалбата и я добавят към кредитите. След това двете полета се запазват в XML файл посредством Update функциите на GameRecovery класа. Накрая се проверява дали е изпълнено условието за бонус игра, като се извика функцията GameController::IsBonusGame(). Ако условието е изпълнено, се инициализира бонус играта, като се подаде m_baseGame като параметър на BonusGame::InitBonusGame().

+ WinFromPaylines():void

Дефинирана на ред 380. Определя печалбата от всички линии, с които е избрал да играе потребителя за текущата игра. Цикъл итерира векторът от линии до брой линии – 1 елемент. Текущата линия се подава като аргумент на WinFromSinglePayline функцията, която връща като резултат печалбата от тази линия. Тази печалба се сумира в локална променлива, държаща печалбите от всички линии. След приключване на цикъла общата печалба от избраните линии се присвоява на полето m_iWin на модела, посредством неговия set метод.

- WinFromSinglePayline(payline: const Payline&): int

Дефинирана на ред 400. Приема като параметър структура от тип линия, която съдържа масив от пет елемента, по един за всяка фигура. Изчислява печалбата на линията, като намери максималната поредица от последователни фигури. За най-добра фигура се взима първият елемент от масива и най-добрата текуща последователност е 1. След това цикъл с 4 итерации, определя най-дългата последователност по следния начин: Ако текущата фигура съвпада с предишната, нейната последователност се инкрементира, ако не – значи се среща за пръв път и последователността се инициализира с 1. Следващата проверка определя дали текущата последователност е по-голяма или равна на максималната. При изпълнение на това условие текущата фигура става най-добра, както и нейната последователност.

След изпълнение на цикъла най-добрата фигура и нейната последователност се подават като аргументи на FigureCoefficient функцията, която връща като резултат коефициента на фигурата. Печалбата от линията се изчислява като продукта от залога за линия и коефициента на фигура. Тази печалба се връща като резултат от функцията.

- FigureCoefficient(figure: const Figures&, iOccurrence: int): int

Дефинирана на ред 446. Параметрите са фигура и броят на нейните срещания. Използвайки switch-case оператор, се изчислява теглото на фигурата, като, ако се среща 3 пъти, теглото е 30, при 4 пъти – 80, а при 5 пъти – 150. Коефициентът се изчислява като продукта от номера на фигурата и нейното тегло. След това функцията го връща като резултат.

- AddWinToCredits():void

Дефинирана на ред 481. Тялото се изпълнява само, ако печалбата е положителна. Взимат се текущите кредити и текущата печалба от модела. Сумират се в локална променлива, като след това тя се подава като аргумент на кредит-мутатора на модела.

+ IsBonusGame():bool

Дефинирана на ред 495. Проверява дали е изпълнено условието за бонус игра, като подава m_iBonusCounter като аргумент на функцията IsBonusGame на класа BonusGame, която проверява дали броячът е равен на 5 (необходимо условие за бонус игра). Ако да- връща true, иначе – false.

+ BetPerLineAsString() const: string

Дефинирана на ред 547. Използва потребителска функция itos, която приема като параметър целочислен тип и ,използвайки stringstream поток, го връща като символен низ. Връща залога за линия като символен низ.

+ CreditsAsString() const: string

Дефинирана на ред 553. Връща кредитите като символен низ.

+ NumberOfLinesAsString() const: string

Дефинирана на ред 559. Връща броя линии като символен низ.

+ TotalBetAsString() const: string

Дефинирана на ред 565. Връща общия залог като символен низ.

+ WinAsString() const: string

Дефинирана на ред 571. Връща печалбата като символен низ.

+ WinningPaylinesAsString() const: string

Дефинирана на ред 577. Използвайки потребителската функция itos(), векторът, държащ номерата на печелившите линии се връща като символен низ.

- ErasePaylines():void

Дефинирана на ред 671. Изтрива съдържанието на вектора държащ линиите, ако не е празен, като използва стандартната функция erase, приемаща като параметри началото и края на вектора. По този начин векторът може да бъде напълнен с линии от нова игра.

BonusGame

Полета

- m_baseGamePtr: GameModel*

Указател към модела, за да може промените върху полетата му, да бъдат отразени и при GameController класа.

- m_iRandCounter: int

Статичен брояч, който да се инкрементира всеки път, когато srand() бъде извикан, за да даде допълнителна случайност на seed.

- m_playerChoice: COLOR

Поле от изброим тип COLOR, който съдържа три стойности: невалиден цвят, черен и червен. Държи избраната боя от играча.

- m_bonusGameResult: COLOR

От същия тип като горното поле. Държи боята, която се е паднала като резултат от текущата игра.

- m_iCredits: int

Държи кредитите от модела.

- m_iBet: int

Залогът за бонус играта. Първоначално това е печалбата от линиите, а при успешно удвояване – удвоената печалба.

- m_iGameWin: int

Държи печалбата от контролера

- m_iBonusWin: int

Полето служи за съхранение на печалбата от бонус играта.

- m_bWonRound1: bool

Променливата определя дали играчът е познал първото удвояване.

- m_iRound: int

Променлива, която се инкрементира при всеки рунд(опит за удвояване)

- m_bQuitBonusGame: bool

Определя дали играчът иска да напусне бонус играта

Член-функции

+ BonusGame()

Дефинирана на ред 43. Конструктор по подразбиране.

+ IsBonusGame(iBonusCounter: int): bool

Дефинирана на ред 49. Определя дали е изпълнено условието за бонус игра. Параметърът е GameController::m_iBonusCounter, който определя броя специални фигури, паднали се на игровите ролки. Функцията връща дали този брой е равен на 5.

+ InitBonusGame(gameModel: GameModel*): void

Дефинирана на ред 55. Параметърът е адресът на модела. Присвоява се на полето m_baseGamePtr. Чрез полето-указател се инициализират полетата за печалба, кредити и залог. На останалите полета се задават стойности по подразбиране.

- SetBonusGameResult(): void

Дефинирана на ред 84. Използвайки библиотечните функции srand() и rand(), се задава произволна боя като резултат от играта m_bonusGameResult.

- UpdateIfWin(): void

Дефинирана на ред 91. Печалбата от бонус играта m_iBonusWin е удвоеният залог. Новата печалба се добавя към кредитите и се присвоява на залога m_iBet. След това се вика методът UpdateWinAndCredit, която обновява печалбата и кредитите на модела, след което ги запазва в XML файл.

- UpdateIfLoss(): void

Дефинирана на ред 106. Печалбата от бонус играта m_iBonusWin е 0. След това се вика методът UpdateWinAndCredit, която обновява печалбата и кредитите на модела, след което ги запазва в XML файл.

+ PlayerSelectedBlack(): void

Дефинирана на ред 120. Играчът е избрал черна боя. Присвоява се eBlack на полето m_playerChoice.

+ PlayerSelectedRed(): void

Дефинирана на ред 125. Играчът е избрал червена боя. Присвоява се eRed на полето m_playerChoice.

- IsValidBet(): bool

Дефинирана на ред 132. Връща дали кредитите m_iCredits е по-голям или равен на залога m_iBet.

- PlayerWon(): bool

Дефинирана на ред 138. Определя дали играчът е спечелил опита за удвояване. Условията са неговият избор m_playerChoice да съвпада с резултата от играта m_bonusGameResult и да не са невалидни бои.

- DoubleUpWins(): void

Дефинирана на ред 147. Инрементира се броячът m_iRound. Две булеви променливи държат условията за първото и второто удвояване на печалбата. Условие за първо удвояване е броячът m_iRound да е равен на 1, а за второ удвояване – броячът да е равен на 2 и да е спечелено предишното удвояване (m_bWonRound1 да е true). Ако едно от двете условия е изпълнено се извиква IsValidBet, която проверява дали общият залог е валиден, т.е. не надвишава кредитите.

Извиква се методът SetBonusGameResult(), която, с помощта на библиотечната функция rand(), задава на случаен принцип червена или черна боя като резултат на текущата игра (полето m_bonusGameResult). След това изважда залогът m_iBet от кредитите m_iCredits.

С условен оператор са обхванати възможностите за печалба и загуба на потребителя. Ако спечели, се вика UpdateIfWin(), който удвоява текущата печалба, обновява кредитите и ги запазва в XML, а при загуба – UpdateIfLoss, който занулява печалбата, обновява кредитите и ги запазва в XML файл.

+ PlayBonusRound(): void

Дефинирана на ред 182. Ако печалбата от модела е положителна, се извиква методът DoubleUpWins().

- UpdateWinAndCredits(): void

Дефинирана на ред 192. m_iBonusWin се предава като параметър на SetIWin метода на m_baseGamePtr, за да обнови печалбата на модела с тази от бонус играта. Полето m_iCredits се предава като параметър на set метода на m_baseGamePtr, за да обнови кредитите на модела. След това запазва двете полета в XML файл, подавайки ги като параметър на функциите UpdateCredits и UpdateWin на класа GameRecovery. Накрая вика функцията SetDefault, който задава стойност по подразбиране на m_playerChoice, m_bonusGameResult и m_iRound.

- SetDefault(): void

Дефинирана на ред 207. Ако m_iRound е равно 1 (потребителят е изиграл един опит за удвояване), полето m_bWonRound1 се инициализира със резултата от функцията PlayerWon(), която връща дали е бил успешен опитът за удвояване на залога. След това се присвояват невалидни бои на полетата m_playerChoice и m_bonusGameResult.

+ GambleAmount(): string

Дефинирана на ред 219. Използвайки потребителска функция itos(), връща залогът от бонус играта m_iBet като символен низ.

+ GambleToWin(): string

Дефинирана на ред 229. Използвайки потребителска функция itos(), връща очакваната печалба (удвоеният залог от бонус играта m_iBet) като символен низ.

+ IsQuitBonusGame(): bool

Дефинирана на ред 238. Връща стойността на полето m_bQuitBonusGame.

+ QuitBonusGame(): void

Дефинирана на ред 244. Ако играчът иска да излезе от бонус играта вика този метод. Занулява полето m_baseGamePtr и задава стойност true на m_bQuitBonusGame.

+ virtual ~BonusGame()

Дефинирана на ред 251. Деструктор.

GameRecovery: - using pugi::xml;

Член функции:

- AddViewToRoot(): void - добавяне на нов възел , към руута на ексемел файла;
- AddVolumeToRoot(): void - добавяне на възела за звук към ексемел файла;
- AddReelsToRoot(): void - добавяне на възела слотовете към ексемел файла;
- AddPaylinesToRoor(): void – добавяне на възел линии към ексемел файла;
- AddPaylinesNumberToRoot(): void- добавяне на възела за броя на линиите към ексемел файла;
- AddBetPerLineToRoot(): void– добавяне на възела за залог за линия към ексемел файла;
- AddTotalBetToRoot(): void– добавяне на възела за общият залог към ексемел файла;
- AddWinToRoot(): void – добавяне на възела за печалба към ексемел файла;
- AddCreditsToRoot(): void – добавяне на възела за кредити към ексемел файла;

- AddBonusGameToRoot(): void – добавяне на възела за бонус игра към ексемел файла;
+ IsSaveGame(): bool – проверка, имаме ли запаметена игра;
+ CreateBlankSave(): bool – създаване на празен документ;
+LoadDoc(docPtr: pugi ::xml_document*): void – зареждане на вече създаден файл;
+Update Doc(docPtr: pugi ::xml_document*) : void – обновяване на .док файла ;
+UpdateView(iView: int): void – запазване на, кое вю е потребителят, чрез брояч;
+UpdateVolume(iVolume: int):void – запазване на кое ниво на звук е потребителя;
+UpdateReels(gameReels: const vector<vector> : vector > &) : void – Запазване на фигурите, които са се паднали на рандом принцип от вектора;
+UpdateNumberOfPaylines(iNumberOfPaylines: int) : void – добавяне на възел , който да запазва броя на линиите;
+UpdateTotalBet(iTotalBet: int): void – добавяне и запаметяване на общият залог към възел на файла;
+UpdateWin(iWini: int) : void – добавяне и запаметяване печалбата към възел на файла;
+UpdateCredits(iCredits: int): void – добавяне и запаметяване на кредита към възел на файла;
+UpdateBonusPLayerChoice(const&:COLOR):void – добавяне и запаметяване на избора на цвят на играча към възел на файла;
+UpdateGameModel(GameModel* const): void – добавяне и запаметяване на текущо състояние на играта;
+LoadView():int – функция за зареждане на ВЮ-то на играта;
+LoadVolume():int – функция за зареждане на звука на играта;
+LoadReels(gameModelPtr: GameModel*): void функция за зареждане на игралните ролки;
+LoadNumberOfPayLines(): int – функция за зареждане на броя на игралните линии;
+ LoadBetPerLine(): void – функция за зареждане на залога за линия;
+LoadTotalBet(): int – функция за зареждане на общият залог;
+LoadWin(): int – функция за зареждане на печалбата;
+ LoadCredits():int – функция за зареждане на кредитите;
+ LoadGameModel(gameModelPtr: GameModel*): void – функция за зареждане на цялостната игра;
+LoadBonusPlayerChoice(): int - функция за зареждане на състоянието на бонус играта;

Инструкции за build на проекта и използвани библиотеки

За да имаме успешен билд на проекта на Windows, са ни нужни следните неща:

1. Задаване на пътя във билд настройките на компилатора, от където добавяме SDL библиотеката: /mingw32/SDL2main/SDL2/SDL2_image/SDL_ttf/SDL2_mixer;
2. След, като сме задали пътя за SDL библиотеката, съответно добавяме иклудите във всеки хедър файл:
    Windows/MacOS:
    “SDL2/SDL2.h”
    “SDL2/SDL_image.h”
    “SDL2/SDL_ttf.h”
    “SDL2/SDL_mixer.h”
    Linux:
    “SDL2\SDL2.h”
    “SDL2\SDL_image.h”
    “SDL2\SDL_ttf.h”
    “SDL2\SDL_mixer.h”
3. За да има успешен билд, трябва компилатора да поддържа C++ 11 или повече; В случай, че е по-стара версия от 11, трябва да добавите: C/C++ Build –> Settings –> Tool Settings -> Miscellaneous -> -std=c++11;