-

**Х А З А Р Т Н А И Г Р А**

**Р У Л Е Т К А**

Финален Проект

**­**

**Участници:**

Виктор Цонев

Васил Какевски

Петър Ефимов

Росица

**Съдържание:**

1. Описание на играта
2. Основни състояния на играта
3. Модули и екрани на играта
4. Recovery / Възстановяване на играта
5. Имплементация
6. UML диаграма на модела на играта

1. **Описание на играта**

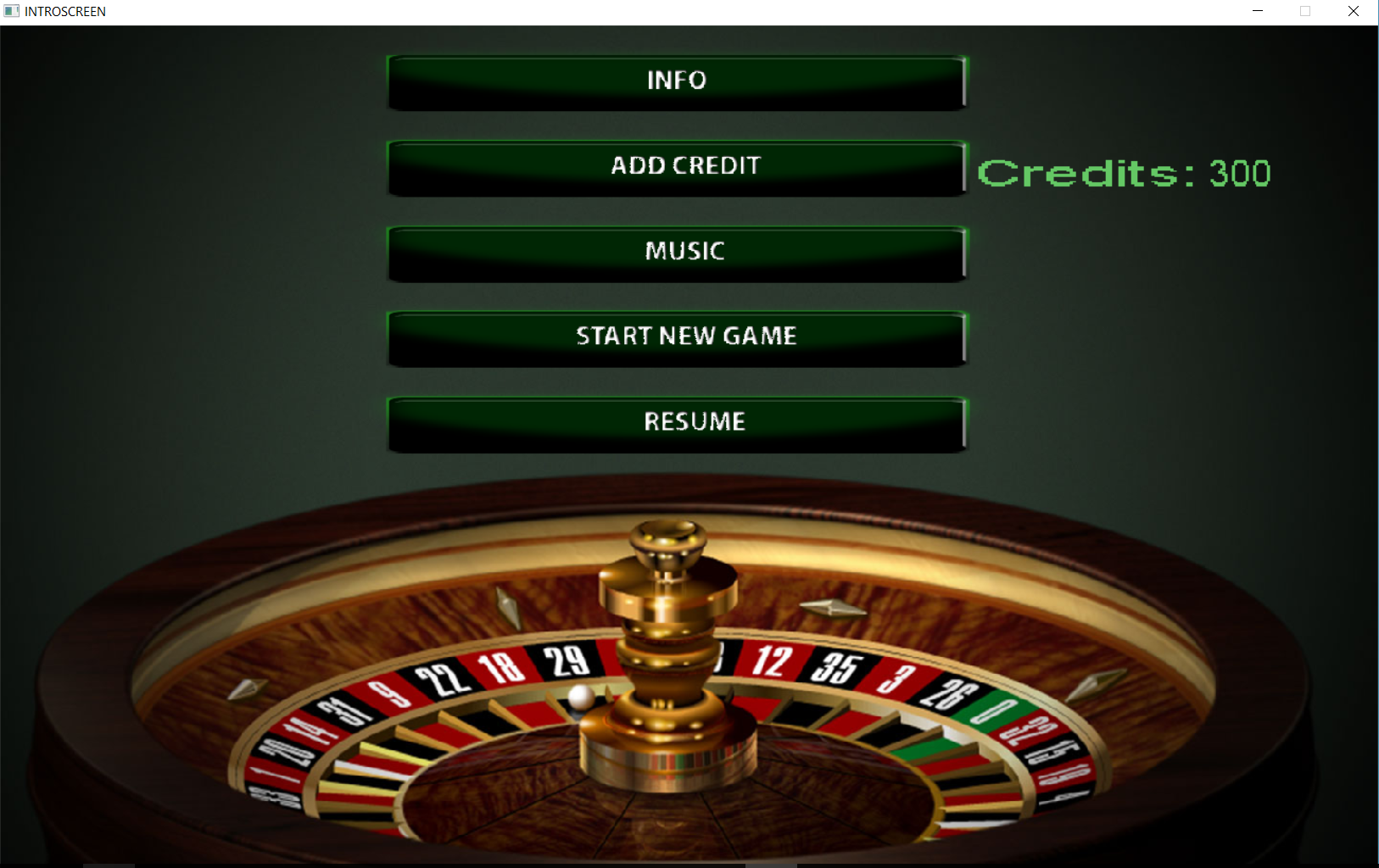
* Хазартна игра, състояща се от различни модули и екрани, бутони, зареждане/настойка на кредити, чипове със залози, въртящо се колело и топче, печалби, история, зареждане от прекъсната игра, записване в XML файл, звуци, текстове и др.
* Целта е участникът да зареди първоначален кредит, с който да прави залози върху игралната маса, след което да завърти рулетката. При приключване въртене на рулетата, топчето се намира на печелившата позиция, обявява се печалбата на играча в отделен екран, като печалбата е изчислена едновремено в кредити и пари.
* Хазартна игра Рулетка за един играч. Всички автоматизирани действия се извършват от компютъра.
* В екрана за залагания, играчът може да залага желани пулове върху избрани числа, както и да избира типа (четни/нечетни) числа или да заложи върху цвят (червен/черен) на числото.
* Позиции за залог на рулетката:
  + 36 числа [1 – 36]
  + 0 - Нулата
  + Четно/нечетно число
  + Черно/червено число

1. **Основни състояния на играта**

Играта Рулетка се състои от пет основни състояния:

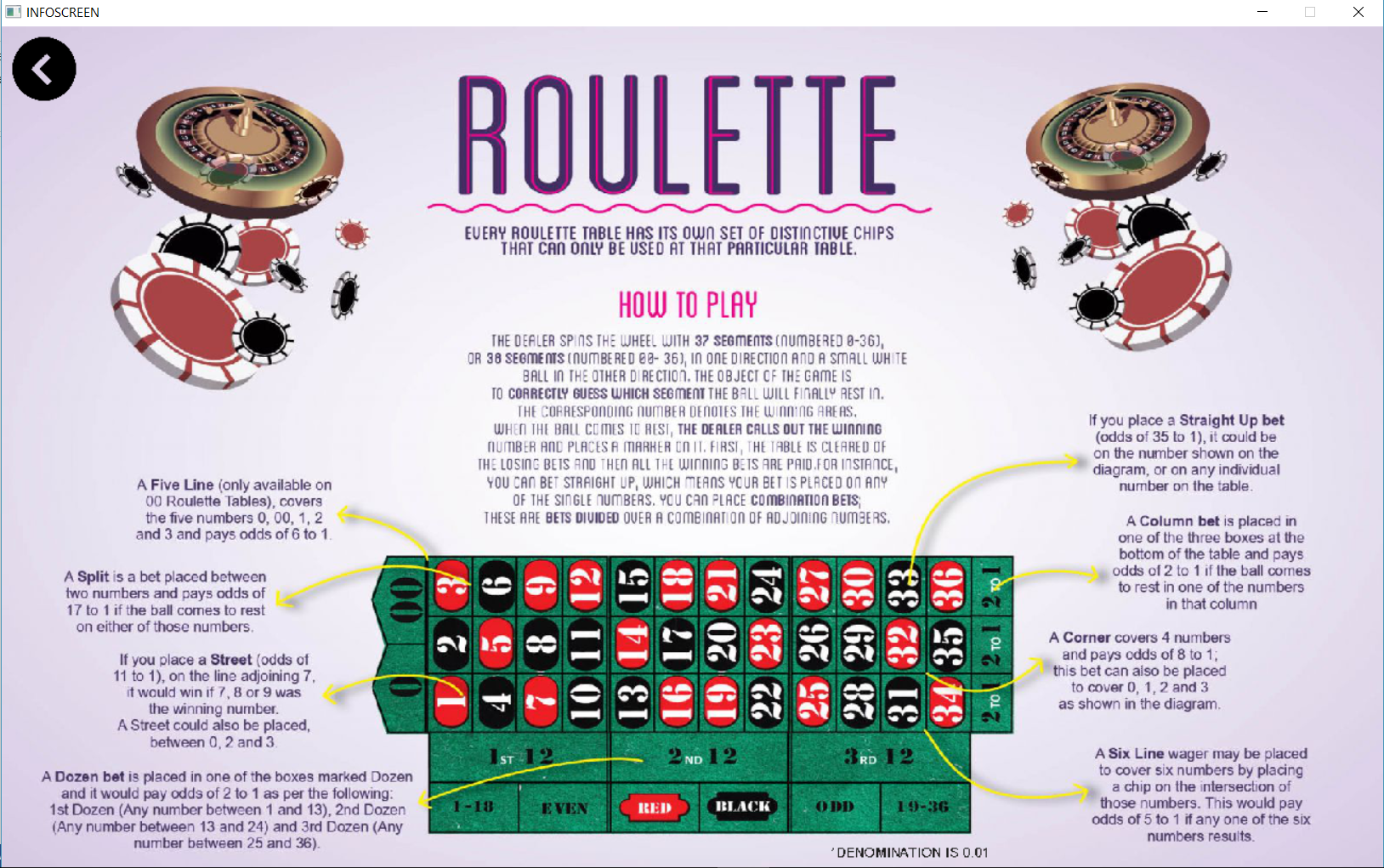
**Състояние Intro**– начално състояние на играта

Представлява начален екран с бутони, които препращат към други модули/екрани/състояния или настойки.



Началният екран Intro има в себе си следните бутони(препратки към екрани):

* **Бутон INFO** - препраща към нов екран с кратка информация за правилата на играта, използваната деноминация за изчисляване на пари спрямо кредити, както и използваната валута.



* **Бутон ADD CREDIT** – работи като вид настройка, и визуализира на екрана стойността на добавените кредити.

При всяко натискане на бутонa, се добавят по 100 кредита към стартиращата сума на играча. Кредитите се изчисляват по определен коефициент/деноминация: 0.01 .

Изпъленена е функционалността зададена в Заданието на играта. Рулетка, а именно: избрана е деноминация 0.01, при което ако се вкарат 100 кредита от бутона Add Credit и след това се натисне бутона Cashout, сумата, която ще се изпише на поздравителния екран ще бъде 1 BGN

* **Бутон MUSIC –** TODO…
* **Бутон START NEW GAME** – бутонът задейства отварянето на нов екран – игралната маса на играта. И играта влиза в ново състояние - „Game“ . В това състояние се нулират всички текущо запазени настройки в играта като се стартира нова игра и записването във XML файл с цел възстановяване от неочаквано състояние започва на ново.
* **Бутон RESUME** - препраща към екрана със играланата маса, влиза в състояние Game и зарежда и визулализира всички залози на играча от момента преди прекъсването. Играчът може да продължи своята игра.

Функционалността е резлизирана със xml файлове и структури данни и е описана в секция “4. Recovery”.

**Състояние Game**– основно състояние със същинската логика на хазарната игра Рулетка. На самостоятелен екран се визуализира зелена играна маса, със числа и други възможни позиции за залози. Играчът има избор на чипове с различна стойност и цвят.

На зелена фон, има числата от 0 до 36 и възможност за залагане и на друг тип позиции, освен конкретни числа, например: четно , нечетно или по цвят. Играчът има избор на залог от 5 различни пула, със стойности: 2, 10, 20, 50 и 100.

Зададения кредит се вижда на екрана, изчислен във сумата/пари/BGN. На базата на кредитa, играчът може да избира и реди/залага пулове на масата, докато има наличен кредит по-голям от 0.

При приключване на залаганията, се очаква играчът да натисне бутона „SPIN“, при което се отваря нов прозорец, където се стартира анимация – завъртане на два обекта: рулетка и топче.

След финално спиране на топчето в определена позиция спрямо число се показва печалбата в нов екран, който след няколк осекунди се скрива и играчът е препратен към екрана със залаганията, за да продължи своята игра.

Общата сума на залога върху всяка позиция/число се визуализира в помощта на SDL\_ttf библиотека. Пример: ако върху избрано число се заложи един пул от 2 кредита и друг от 10 кредита, резултатът който ще се покаже на изгралната маса в/у това число ще бъде 12.

На екрана GameBoard има множество налични бутони, описани детайлно в секция „3.Модули и екрани“ . Те предлагат вид настройка или влизане в други състояния на играта: има възможност за проследяване на историят, accounting информация, Cashout бутон – взимане на парите и напускане на играта, завъртане на рулетката и изчистване на залозите.

**Състояние Win**– състояние на играта при печалба.

В това състояние на играта се влиза, ако залогът е по-малък

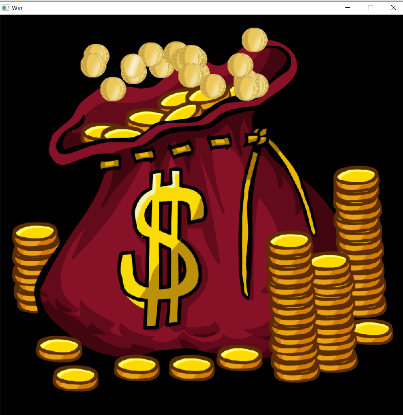
от печалбата. Тогава се появява нов екран с анимация, която показва

на играча, че печели. Екранът стои активен 6 секунди

и след това зарежда игралната маса на рулетката, за да може

играчът да продължи своята игра.

Тодо: се изпише спечелената сума в кредити и пари едновременно



**Състояние Bonus**– състояние на допълнителна печалба

В това състояние се влиза след всеки 20 завъртания на рулетката.

Играчът получава бонус кредити, които зависят от големината на залозите или по точно - 13% от всички залози независимо дали е имало печалба или не. Преди двадесетото завъртане на рулетката след залога се появява нов екран, на който е изписана сумата, която играчът получава като бонус мистериозна печалба.

**Състояние Outro**– влиза се по желание на играча, след натискане на бутона Cash Out. Така се изплащат текущите пари на играча.

След натискането на бутона Cash Out се появява нов екран, с изписано поздравление за играта и деноминирана сума, която печели учасникът в български лева. Екранът стои активен 10 секунди, след което автоматично се прехвърля към екрана Intro.

1. **Модули и екрани на играта**

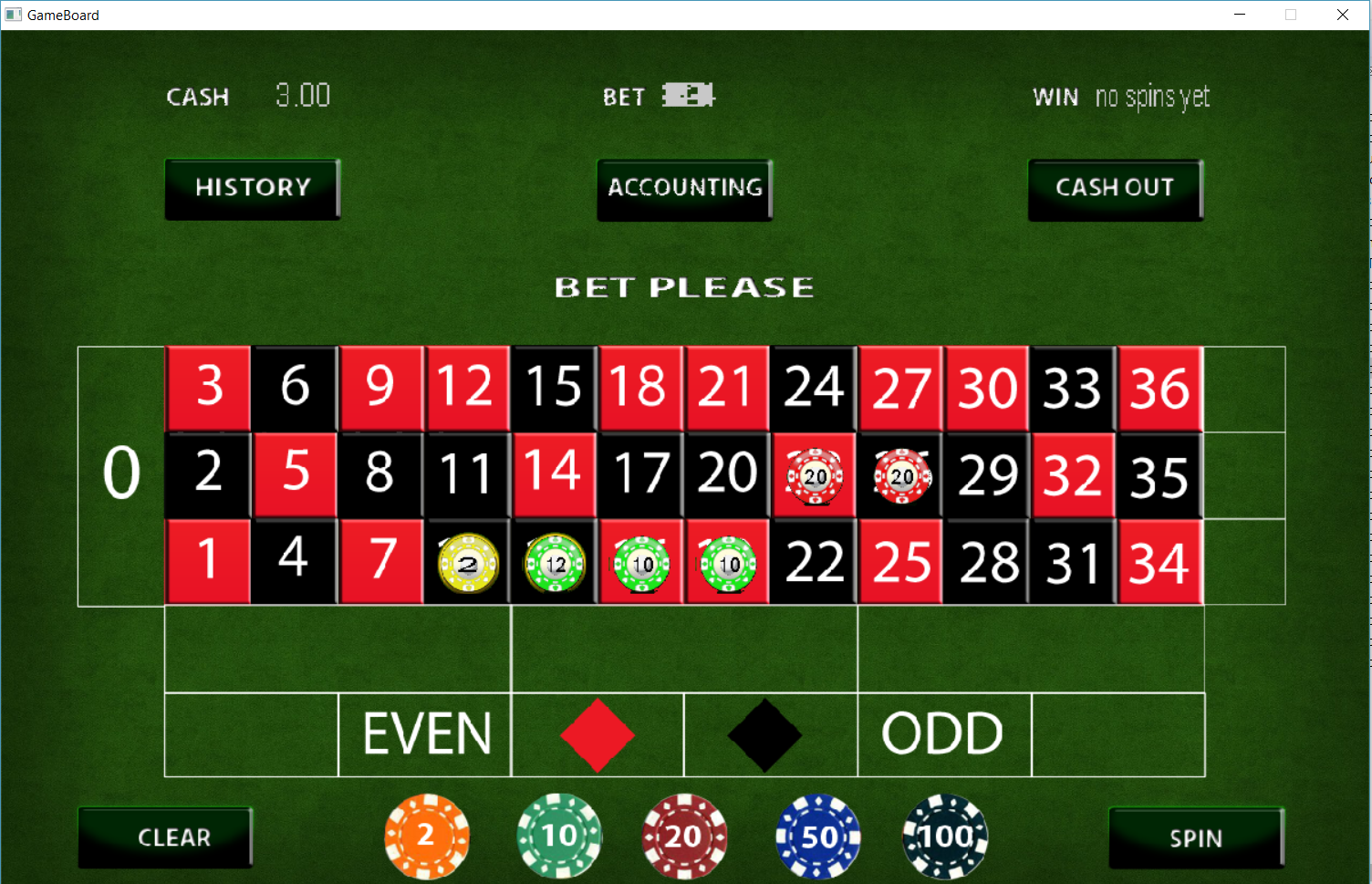
**Екрани:**

* Задаване на начален кредит – става чрез натискане на бутона за настройка на кредит в Началния Екран на играта.



* Модули в състояние Game (екран GameBoard)

В състояние ‘Game’, на екрана ‘GameBoard’ има различни модули: кредити, бутони, тапи, история, въртящо се колело с топче.



* модул с кредити - показва стойностa на текущите кредити, стойността на заложените кредити по тапито (ако има такива) и печалбата, ако играта е печеливша



* модул с различни бутони:
  + бутон за смяна стойността и вида на чипа



* + бутон за изчистване на залозите по тапито



* + бутони за залагане на червено/черно



* + четно и нечетно

* + бутон за показване на стрaница счетоводство/accounting

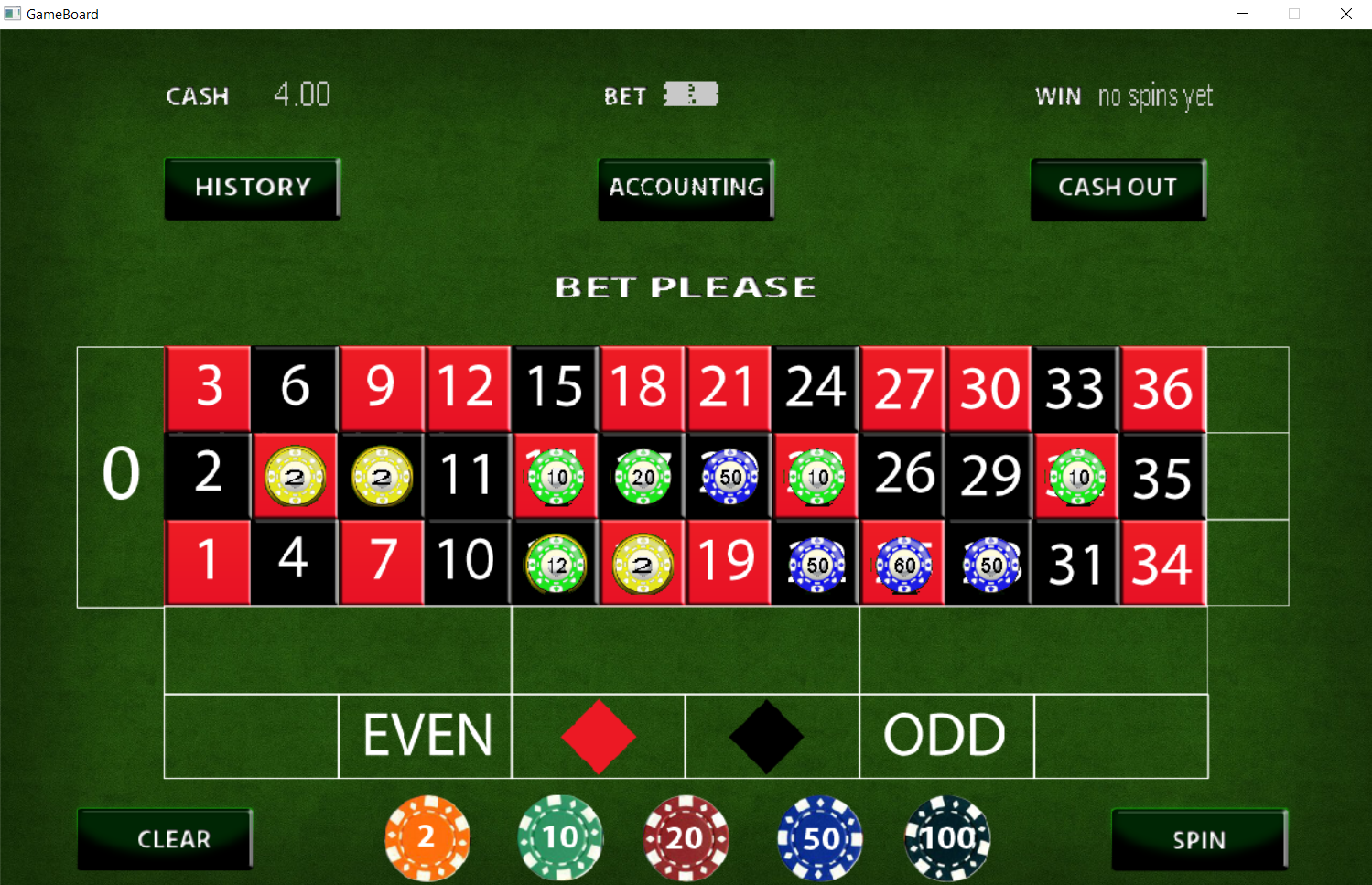


TODO

* + бутон за край на залози и завъртане на рулетката



* модул тапи - игралната маса със сектори от 0 до 36. Залаганията могат да бъдат само на целите числа. Не може да се залага меджду две, три или четири числа с един чип. При кликане някъде из определен квадрант/клетка, пулът се позиционира автоматично централно.



* модула с история - показва последните 18 печеливши паднали се числа



* модул с въртящото се колело – обект разделен на равни части, съдържащ числата от 0 до 36, който се върти в една посока, а топчето в обратната посока докато не се падне някое число



ТОДО: ...

1. **Recovery / Възстановяване на играта**

Възтановяване на играта от неочаквано прекъсване е функционалност, реализирана с помощта на XML файлове. Използвани са библиотеките pugixml. При всеки залог/ход на играчата, необходимата информация се записва в xml файл.

Логиката е реалзизирана в самостоятелен метод на класа Application.

На метода за писане в XML, като аргумент се подава ( map<int, int> ), който има за ключ - номера на клетката, а за стойност - стойността на залога. Обхождайки map-а, се записва в XML текущото състояние на залозите по дъската заедно с всичката необходима информация за тях.

При натискане на бутона Resume в началния екран, играта се възтановява от момента на прекъсването. На екрана GameBoard са възтановени и визуализирани всички залози (и техните стойности), за да може играчът да продължи да играе от момента на прекъсването на играта.

Примерен код в XML-а:

<?xml version="1.0"?>

<Bet cell="0" amount="2" />

<Bet cell="1" amount="0" />

<Bet cell="2" amount="6" />

<Bet cell="3" amount="12" />

<Bet cell="4" amount="30" />

<Bet cell="5" amount="10" />

//ТОДО: обясни resume button-a технически

1. **Имплементация**

**Класове в играта:**

**Class Application**

Класът Application е основния клас, който управлява различните състояния, модули и настройки на играта.

Този клас управлява отделните екрани и поддръжката на четене/писане от XML. Изчислява залозите, завъртането на рулетката, печалбата, бонус печалбата, историята на печелившите числа...

ТОДО:

Методи на класа:

**void** **initIntro**(); - препраща към началния екран за избор на модул/екран

**void** **initInfo**(); - препраща към екран с информация и правила на играта

**void** **initGameBoard**(); - препраща към екран с игрална маса, чипове за залания, рулетка и е свързан с основната логика

**void** **initRoulette**(); - препраща към екран с въртяща рулетка с топче

**void** **initOutro**();

**void** **initWin**();

**void** **initSpin**();

**void** **Free**(); - освобождава ресурсите ( освобождава паметта на създадените с new обекти и др.)

**void** **GamePlay**();

**void** **DisplayBets**(**int** x , **int** y , **int** color, vector<Point> v\_allBetPoints); - рисува/позиционира заложените пулове върху позволените числа/позиции

**int** **CalcQuadrandClicked**(**int** x, **int** y); - спрямо координатите в/у които кликнато с мишката, изчислява поредността на клетките/числата започвайки да ги брой по ред и колона и функцията връща число, което е част от редицата [0-43]. На базата на това число се определя къде е направил залога си играчът.

**void** **SpinRoulette**(LTexture &ball,**int**); - ТОДО

**int** **spinBall**();- ТОДО

**bool** **WinAnimation**();- метод с който при печалба по голяма от заложеното показва анимирани златни монети

върху Win екрана.

**virtual** **~Application**();-

GameState **getMenuState**() **const**; - ТОДО

**void** **appendToXML**(map<**int**, **int**> ); //bets

**void** **appendToXMLHistory**(queue<**int**>); //history

map<**int**, **int**> **readXMLWriteMap**(string pathXml);

queue<**int**> **readXMLWriteQueue**(string pathXml);

**Class Background**

Помощен клас, който отговаря за визуализацията.

Има мембъри, които се грижат за зaдаване и четенете на височината, ширината, името на прозореца и пътят с името на файла, които ще се зареди.

Има методи за зареждане на файлове от тип картинка в прозорци, рендериране и тяхното почистване.

Методи на класа:

**Background** (string title= "",**int** width=0, **int** height=0,string path=""); -конструктора на класа;

**virtual** **~Background**(); - деструктора на каса;

**void** **Clear**(); - метод който освобождава и занулява surface ;

**void** **Show**(); - метод който се грижи за визуализирането ;

**bool** **loadMedia**(); - метод който се грижи за зареждането на картинките ;

**bool** **Init**(); -

**int** **getHeight**() **const**;

**void** **setHeight**(**int** height);

**const** string& **getPath**() **const**;

**void** **setPath**(**const** string& path);

**int** **getWidth**() **const**;

**void** **setWidth**(**int** width);

**void** **setTitle**(**const** string& title);

**Class Ball**

Помощен клас, който отговаря за визуализацията.

Клас – представляващ обект със координати x, y и скорост на движение. Функции, показват или задават информация за скоростта на движение, и за това дали е спряло има отделен метод.

Методи на класа:

**Ball**(**int** \_x,**int** \_y ,**int** \_speed);

**virtual** **~Ball**();

**bool** **spin**();

**bool** **isBallStoped**(){**return** isStoped;}

**int** **getSpeed**() **const**;

**void** **setSpeed**(**int** speed);

**Class Button**

Помощен клас, който отговаря за визуализацията на отделни обекти тип Бутони.

Наследява LTexture обекта и дефинира допълнителни методи за: дали е кликнато в/у обекта, дали е Hover-нато в/у обекта.

Методи на класа:

**Button**(**int** x = 0, **int** y = 0);

**virtual** **bool** **isHover**();

**virtual** **bool** **isClicked**( SDL\_Event \*);

**virtual** **~Button**();

**Class Credits**

Помощен клас при изчисляване на кредитите и залозите на играча.

Има логика за залагане, промяна на кредити, информация за залога, печалбата, наличния кредит и дали е прибрал играчът своята печалба.

Методи на класа:

**Credits**(**int** = 0);

**Credits**(Credits &);

**virtual** **~Credits**();

**void** **AddBet**(**int**);

**void** **ChangeCredits**(**int**);

**void** **SetBet**(**int**); //needless

**int** **GetBet**() **const**;

**int** **GetCredit**() **const**;

**int** **GetPossibleWin**() **const**;

**void** **SetPossibleWin**(**int**);

**bool** **isCashedOut**();

**Class Globals**

Клас, съдържащ константни или статични промениливи.

Пример:

**const** **int** PULLS\_W = 60;

**const** **int** PULLS\_H = 60;

**const** **int** NUMBER\_OF\_SECTORS = 37;

**const** **int** MULTIPLIER\_NUMBER = 36;

**const** **int** MULTIPLIER\_COLOR = 2;

**const** **int** MULTIPLIER\_PARITY = 2;

**Class LTexture**

Клас служещ за вузуализация (на текстури/обекти/картинки).

Класът има връзка със class Background. Отделните инстанции на класа LTexture се рендерират/визуализират с помощта на Renderer обект, част от класа Background.

LTexture предствавлява обект, който има ширина, височила, координати x,y. Методи за техната промяна и задаване на стойсности. Функции за лоудване на картинки, рендерването им и освобождаване на ненужните ресурси.

LTexture има базовата фунцкионалност за показване на картинки, която се наследява от множество други класове:

Ball, RouletteWheel, Button.

Методи на класа:

**LTexture**(**int** x = 0, **int** y = 0);

**virtual** **bool** **loadFromFile**(SDL\_Renderer\* gRenderer, std::string path,

**int** width = -1,

**int** height = -1);

**virtual** **~LTexture**();

**virtual** **void** **render**(SDL\_Renderer \* gRenderer, SDL\_Rect \*clip,

**double** angle = 0,

**int** w = 0, **int** h = 0);

**virtual** **void** **free**();

**virtual** **void** **Show**();

**int** **getHeight**() **const**;

**void** **setHeight**(**int** height);

**int** **getWidth**() **const**;

**void** **setWidth**(**int** width);

**int** **getX**() **const**;

**void** **setX**(**int** x);

**int** **getY**() **const**;

**void** **setY**(**int** y);

**virtual** **void** **setPosition**(**int** \_x, **int** \_y);

**Class Main**

Извиква обект от тип Application, който се явява гръбнакът на играта.

**Class Point**

Класс с 4 члена:

* координатът на кликване с мишката
* координатът на кликване с мишката
* цвета на заложения пул
* стойността на заложения пул: 2,10,20,50 или 100

ДОТУК

**Class Pools**

Наследява Button клас-а.

Отделно има като композиция обект от класа Credits.

Методи на класа:

**Pools**();

Credits cr;

**Pools**(Credits& ,**int** x=0,**int** y=0,**int** number=0);

**virtual** **~Pools**();

**bool** **isBet**();

**int** **GetNumber**();

**int** **PossibleWin**();

**Class RouletteWheel**

Кратко описание: ТОДО...

Методи на класа:

**RouletteWheel**(**int** \_x,**int** \_y,**double** angleSpeed);

**virtual** **~RouletteWheel**();

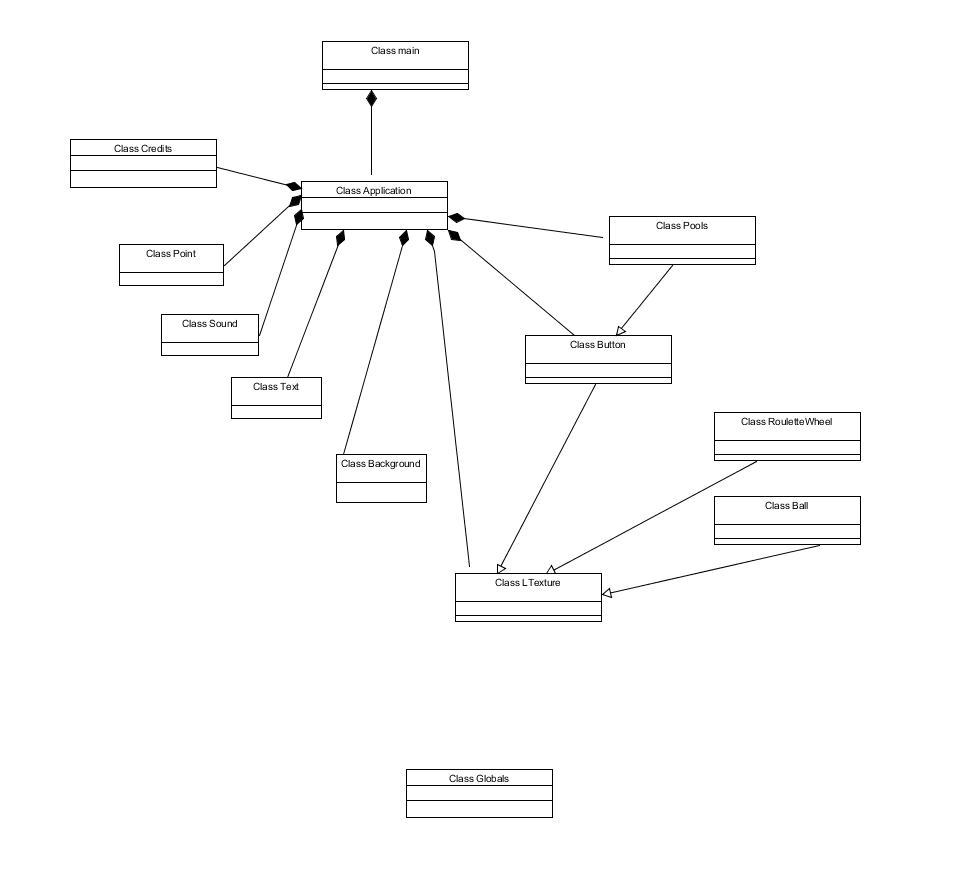
**void** **Spin**();

**Class Sound**

**Class Text**

1. **UML диаграма на модела на играта**

ТОДО: С помощта на UML tool да се създаде визуално създадения модел на играта. Ако визуално има място , да се добавят properties and methods на класовете.



\* Volume – бутон за регулиране звука на играта

Outro – трябва да се покаже поздравителен екран със спечелената сума, конвертирана от

кредити в пари, според избраната деноминация, като този екран трябва да остане 10 секунди и

след това да се премине автоматично към екран Intro.

\* Счетоводство/accounting - на тази допълнителна страница трябва да се покаже

информация каква е била печалбата от всяко число поотделно и сбора от всички печалби до

момента, както и общата сума пари вкарани чрез бутона Insert Credit, общата сума пари

изкарани с бутона Cashout и какъв е процента на възвръщаемост RTP – return to player)

\*\*\* Всеки проект трябва да съдържа следните компоненти:

- Source code.

- Документация – техническа документация на проекта, схеми, диаграми и др.

- Инструкции за билдване на проекта, списък на външни библиотеки и т.н.