Документација за играта Спринг Нинџа

по предметот Визуелно програмирање

Изработиле:

Христијан Антов

Марија Попова

Скопје,мај 2015

1. **Општ опис на играта**

Играта Спринг Нинџа е адиктивна игра во која главна цел ни е постигнување на што поголем резултат(Score). За да постигнеме што поголем score потребно е да скокнеме што е можно повеќе платформи вкупно ,или што е можно повеќе платформи во еден скок . Во спротивно ако не успееме, играта за нас ќе заврши. Всушност тоа ја прави оваа игра предизвикувачка и “заразна”.Играта која е имплементирана е рудиментална копија од следнава игра:

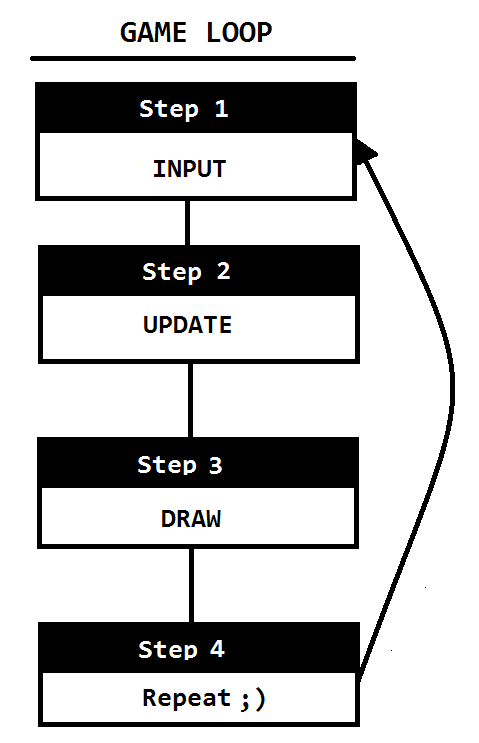
<https://play.google.com/store/apps/details?id=com.ketchapp.springninja&hl=en>

Спринг Нинџа е многу лесна за играње, затоа што со едно едноставно копче “space” од тастатурата правиме се. Со кликање на копчето и соодветно држење на истото, ние ќе скокнеме толку колку што сме го задржале. Нормално имаме прогрес бар кое ни служи за ориентација колку веќе го држиме копчето. За да постигнеме што подобри резултати на играта треба да направиме едукативна претпоставка за тоа колку да го држиме копчето и тоа е се.

1. **Начин на имплементирање на играта**

Како и секоја игра од програмерска перспектива , па така и нашата користи **Game Loop** концепт со цел да ја поедноставни комплексноста при имплементација и правилно да го концептуализира животниот циклус на играта.

**Game Loop-от за нашата игра изгледа вака:**

****

* **Организација на класите и процесот на исцртување**

Сите класи кои што претставуваат компонента која треба да се исцрта, наследуваат од апстрактната класа SceneObjects.

Ова ни дозволува да ги чуваме било кои објекти од различните класи кои тековно треба да се исцртаат во единствена генеричка листа :

public List<SceneObjects> DrawableList

Таквите класи се : Platform , Ninja , Menu.

Цртањето се изведува врз GamePanel објект наречен **Stage**

(GamePanel е всушност Panel меѓутоа наша класа која наследува од Panel со цел да го наместиме DoubleBuffered својството на true , бидејќи е *protected*) .

Процесот на исцртување се одвива на наједноставен начин во :

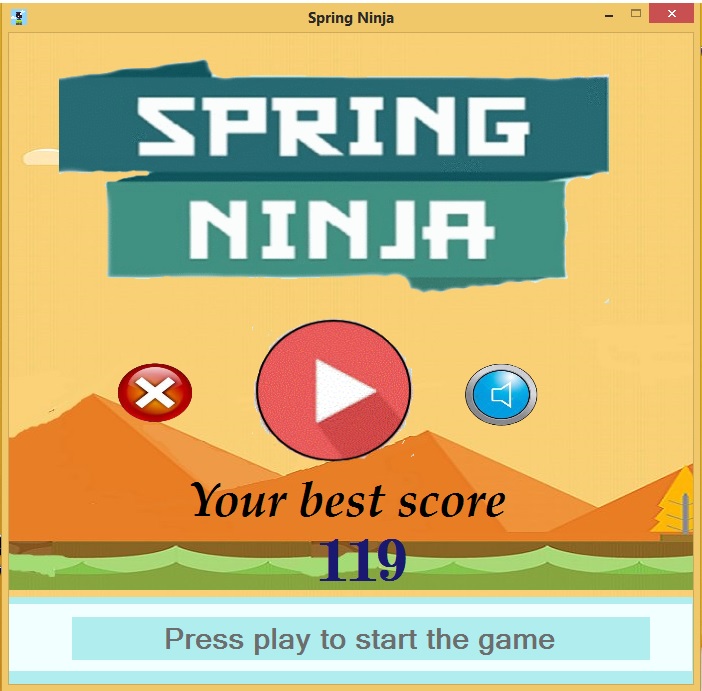
private void Stage\_Paint(object sender, PaintEventArgs e)

Хендлерот на тој начин што го повикуваме Draw(Graphics g) препокриениот метод за сите објекти кој што тековно се наоѓаат во DrawableList генеричката листа.

* **Два клучни методи во кои се врши комплетна замена на приказот на играта.**

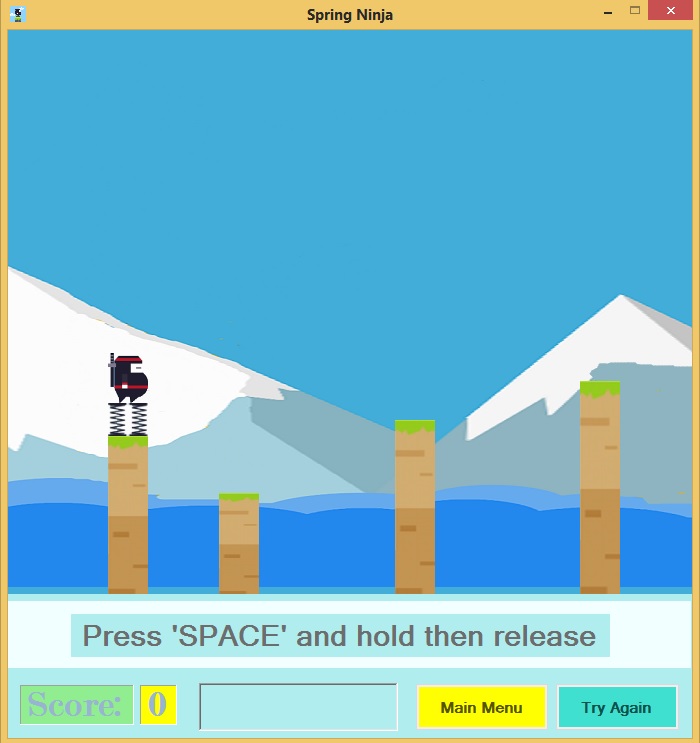
public void InitStartMenu()

Овој метод се повикува при **старт на играта** и при секое кликање на **btnMenu** копчето. По повикување на овој метод играта изгледа вака :



public void InitStartGame() Овој метод се повикува при клик на **imagePlay** компонентата или пак при клик на **btnNewGame** копчето.

1. Во овој метод се иницијализираат 101 инстанци од класата Platform со случајна висина , случајна слика и случајна дистанца помеѓу себе.
2. По завршувањето на тој процес се иницијализира **Ninja ninja** објектот со позиција над првата платформа од горните .
3. Се полни генеричката листа **DrawableList** со соодветните инстанци.
4. Вака би можело да изгледа:



**3.Опис на класата Ninja**

Класата **Ninja** пред се , наследува од апстрaкната класа **SceneObjects.**

*Како основни променливи во класата се чуваат:*

1. Image ninja 🡪 Се црта над платформата или во воздух при скок.
2. Image imageSpring 🡪 Овде се чува Ресурсот од сликата со федерот(springot):(*Properties.Resources.spring)*
3. Image imageNinja 🡪 Овде се чува Ресурсот од сликата со нинџата:(*Properties.Resources.ninja)*
4. bool isJumpingImage 🡪 Ја чува состојбата дали нинџата скока во моментот или не.
5. bool Stop\_Horizontal\_Movement 🡪 Бидејќи при судар на нинџата со платформата странично по законите на физиката нинџата треба да запре не движејќи се “внатре” во платформата по x-оската, затоа оваа bool променлива ја чува состојбата дали треба да се запре хоризонталното движење на нинџата или не .
6. bool First\_Time 🡪 Оваа променлива се користи со цел да знаеме дали нинџата за првпат ја напушта платформата при скок.
7. int SpringHeight 🡪 Оваа променлива се користи со цел да го анимираме протегањето како и собирањето на imageSpring сликата , зголемувајќи ја и намалувајќи ја нејзината висина соодветно.Иницијално таа има вредност 13.
8. int height 🡪 Висината на нинџата.
9. int X 🡪 X-координатата која се наследува од SceneObjects.
10. int Y 🡪 Y-координатата која се наследува од SceneObjects.
11. int dx 🡪 забрзувањето по X-оската при скок .
12. int dy 🡪 забрзувањето по Y-оската при скок .

**Методи во класата Ninja:**

1. public override void Draw(Graphics g) 🡪 Овој метод го препокрива Draw() методот од апстрактната класа и го исцртува Ninjata на соодветната позиција со соодветните ресурси.
2. public void ChangePicture() 🡪 Овој метод ја туглира сликата на Ninjata при состојба на скок или на мирување најизменично.
3. public void AnimateReflection(int reflection)🡪 Овој метод ја намалува или зголемува висината на Ninjata за вредноста на reflection параметарот.
4. public void Jump(GamePanel stage ,SpringNinja form) 🡪 Во овој метод се имплементира физиката и логиката на Ninjata при скокот (забрзувањето за dx , dy ) и го лизга панелот Stage на кој всушност цртаме.