|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| ПРОГРАМИРАње видео игре ‘THE CHICKEN RUNNER’ | | | |
| РАЧУНАРИ | | | |
|  | |
| Кандидат |  | | Ментор |
| ненад гвозденац |  | | имре герштмајер |
|  | Сомбор, 2021. године | |  |

Садржај

[Садржај 2](#_Toc73626514)

[Задатак матурског рада 3](#_Toc73626515)

[1. Увод 4](#_Toc73626516)

[Опис проблема 6](#_Toc73626517)

[1.1. Циљ рада 6](#_Toc73626518)

[2. Теоријске основе 7](#_Toc73626519)

[2.1. Unity 7](#_Toc73626520)

[2.2. Visual Studio 2019 8](#_Toc73626521)

[2.3. C# 10](#_Toc73626522)

[2.4. Tween 11](#_Toc73626523)

[3. Решавање проблема 12](#_Toc73626524)

[3.1. Кретање објекта 12](#_Toc73626525)

[3.2. Бирање објекта играча и кретање 13](#_Toc73626526)

[3.3. Скакање и летење птице 15](#_Toc73626527)

[3.4. Ударац у непријатеља и завршетак нивоа 16](#_Toc73626528)

[3.5. Звук и музика 18](#_Toc73626529)

[3.6. Опције 19](#_Toc73626530)

[3.7. Главни мени 21](#_Toc73626531)

[3.8. Победа 23](#_Toc73626532)

[4. Закључак 24](#_Toc73626533)

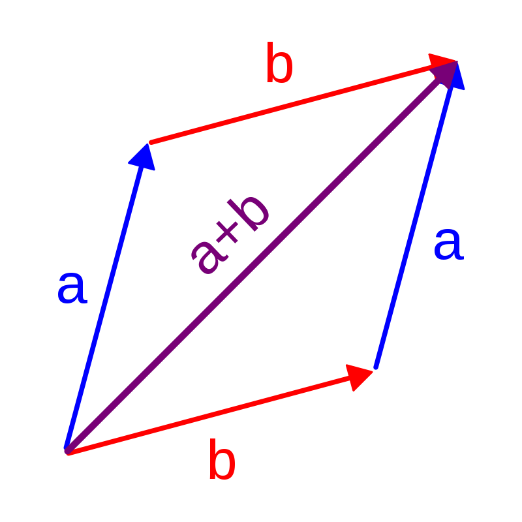
[Литература 25](#_Toc73626534)

Задатак матурског рада

Задатак матурског рада је био да направимо игрицу у Unity-у коришћењем C# језика. Имали смо неколико недеља да научимо основе тог језика као и едитора, распитамо се, те направимо игриву видео игру.

# Увод

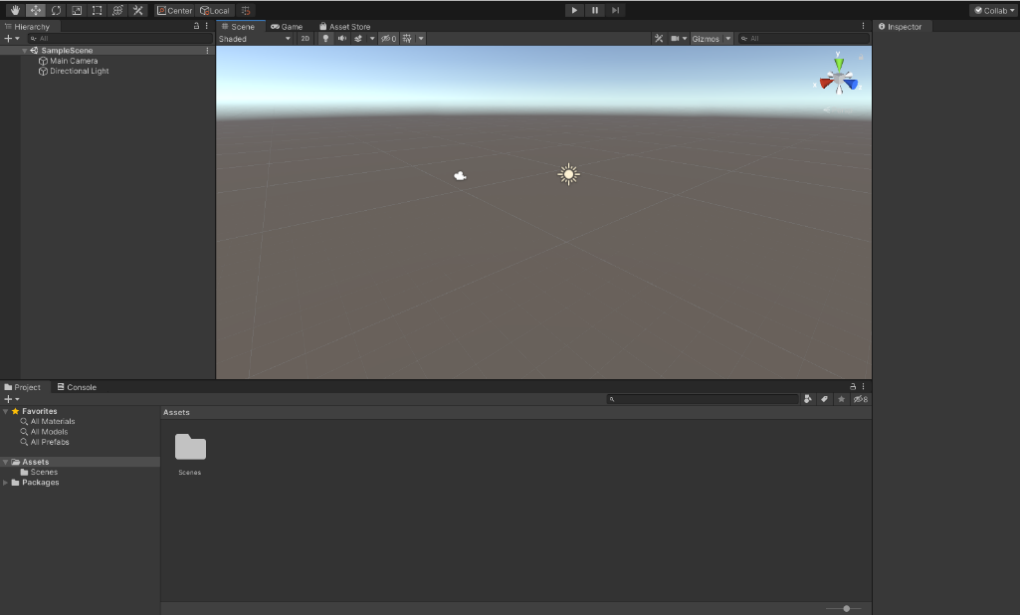
Игрица ‘The Chicken Runner’ је настала приликом рада радионице Лауда, итерације 0. Наиме, ова радионица је направљена да поучи младе средњошколце да уђу у свет креирања видео игрица. Ментори радионице су нас у почетку учили како ради програмски језик C#; а потом полако показивали како се ради у Unity платформи. Најосновније ствари на почетку морали смо научити, a то је како се вектори крећу, када им додајемо вредности бројева на некој од 3 осе (X, Y, Z) и како ће се они понашати.



*Слика 1: Вектори*

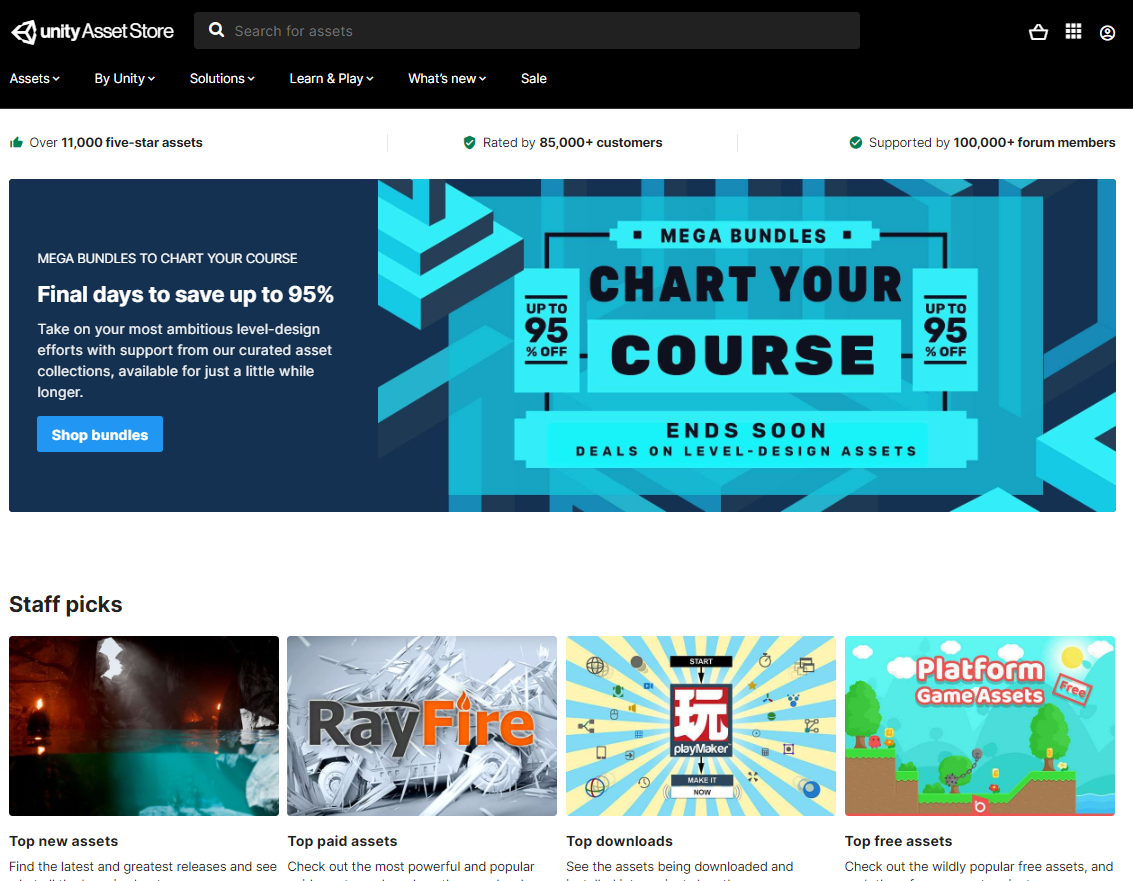
Након што смо научили како се вектори понашају у простору, све видели како ради, прешли смо на учење постављања објеката у Unity програму. Објекти представљају неке материјалне предмете који се постављају у свет који Unity сам генерише. Научили смо како да скинемо неке, већ ручно направљене објекте и поставимо их у свет.

Прво смо креирали нови свет у Unity-у, након му додали име, специфирали локацију где хоћемо да га направимо, и потом поставимо објекте у свет.



*Слика 2: Изглед празног света, почетан екран едитора у Unity*

Ако желимо да убацимо неки објекат, идемо на <https://assetstore.unity.com/> веб-сајт. Након тога, претражимо који објекат желимо, скинемо га, и убацимо у Unity.



*Слика 3: Изглед Asset Store-а*

Ово је основа како се креће учење програмирање видео игара. Након овога, убацујемо то што скинемо, и онда програмирамо даље понашање саме игре.

## Опис проблема

Проблем је био да нађемо саме Asset-e са Asset Store-а, као и само програмирање скрипти које се користе за рад саме игре. Битно је било да те скрипте буду уредне, без компликација, као и да су разумљиве за рад. Било је потребно направити птичицу која се креће према играчевим контролама. Која лети, скаче и заобилази препреке, како би дошла до циља у најкраћем времену.

## Циљ рада

Циљ је да направимо игрицу која као главног играча има птичицу која се креће и заобилази препреке.

# Теоријске основе

## Unity

Unity представља основни едитор за постављање објеката у простор за видео игру.

Unity је вишеплатформски погон игре (Game Engine), са уграђеним програмерским простором (IDE), који је произвео Јунити Технолоџиз. Користи се за развој видео игара за веб плагинове, десктоп платформе, конзоле и мобилне уређаје. Развијен је 2005. године у вишеплатформским покретачима игара.

Unity је платформа која као основу користи језик C#. Све скрипте, као и сам едитор, су покретани од странице тог језика. Као и све остале платформе за програмирање видео игара, Јунити има опцију да повезује играче или објекте са неким ручно-написаним скриптама. Ове скрипте могу утицати на звук, видео, померање, обртање и разне ствари које су повезане са самим објектом.



*Слика 4: Лого Unity платформе*



*Слика 5: Изглед терена у Unity-у*

## Visual Studio 2019

Visual Studio као програмерско окружење (IDE), нам је служило као место где смо писали скрипте које ће касније бити повезане са самим објектима и играчима. Ово програмерско окружење је веома добро направљено и повезано са Unity-јем, јер када повежемо исте, имамо могућност завршења речи које пишемо, као и проверу синтаксе да ли нам је програм у реду.

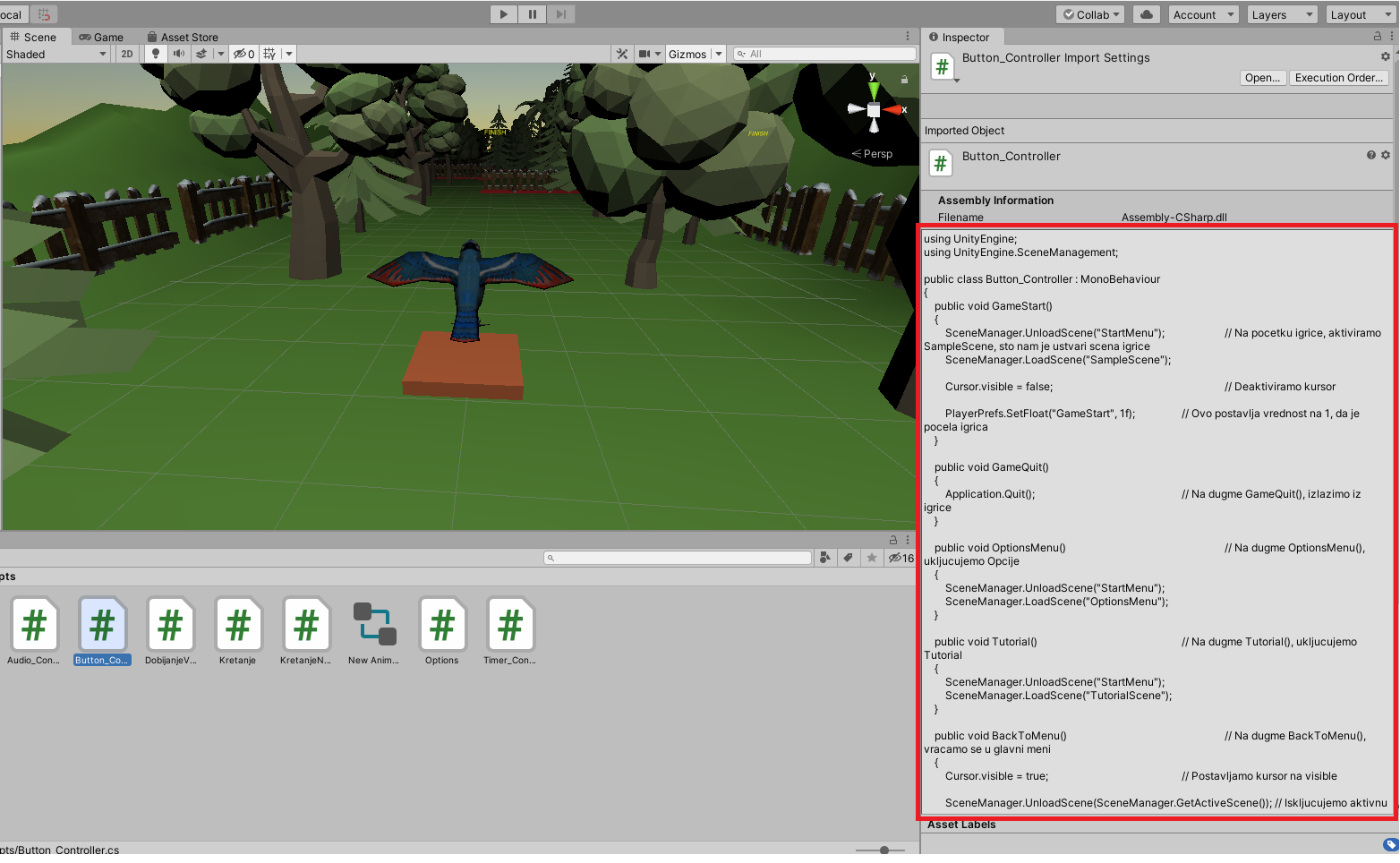


*Слика 6: Изглед кода у Visual Studiјu*



*Слика 7: Лого Visual Studia*

Visual Studio је један од најпознатијих Code Editor-a на данашњем маркету, првенствено због своје брзине и поузданости. Он пружа могућности програмирања у мноштву језика, од C, C++, C#, Ruby, Python-a и шире. Поуздан је из разлога што пружа веома једноставну конфигурацију простора и помагала која служе за помоћ при програмирању. Нама је битан зато што је он веома лако повезан са Unity-јем, на начин на који у самом Unity/u ми видимо сам код, и можемо га компајлирати при покретању видео игре.



*Слика 8: Изглед кода у инспектору*

Као што видимо на слици 8, чак и у Unity Инспектору (Inspector) се види сам код који је написан у Visual Studiјu, што нам даје лаку прегледност самог кода, као и на увид грешке.

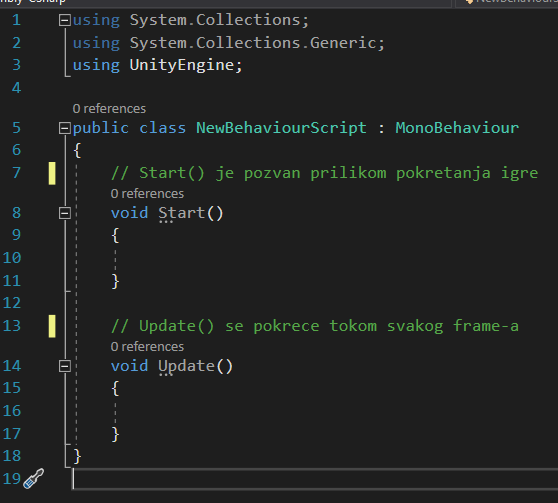
## C#

C# је програмски језик који је направио Микрософт. Ово је програмски језик који се користи за разне ствари, од програмирања веб апликација, десктоп апликација, видео игара, до управљања базама података итд. Настао је 2000. године од стране Микрософт-а као један од делова .NET иницијативе, а 2002. године добио и признање од ISO-a. Дизајниран је од стране Андерс Хеђисберга. Овај језик се гледа као један од Common Language Infrastructures језика, Најновија верзија језика је верзија 9.0, која је и део Visual Studia 2019.

Овај језик је настао на референцу програмског језика C, с назнаком да је вид на томе да ово буде скроз објектно оријентисан језик – „C like Object-Oriented Language”.

Занимљивост је да је сам креатор Јава програмског језика, Џејмс Гослинг, назвао овај језик „имитација јаве“, због сличне синтаксе.

Код нашег случаја, програмирања у Unity engine-у, приликом креирања скрипте, добијамо две аутоматски-генерисане методе (или функције) за рад са програмом – Start() и Update().



*Слика 9: Нова направљена скрипта у C#*

Start() се дешава само једном, на почетку видео игре, и више се не дешава осим ако није позвана мануелно.

Update() се репетира приликом преласка на сваки следећи фрејм (frame).

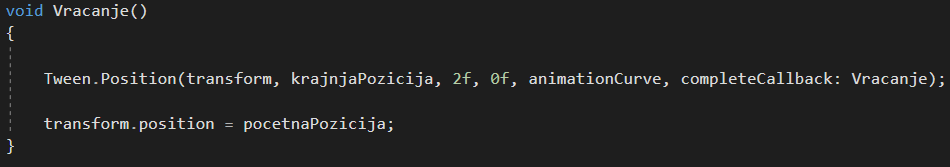
## Tween

Tween библиотека је једна од многих корисничко интегрисаних библиотека за програмирање покрета у C#. Ја сам је користио за покретање птица непријатеља који представљају препреке на које играч не сме да налети. Помоћу Tween-а можемо да направимо кретање на начин на који не бисмо могли путем обичног додавања гравитације, гурања објеката итд.

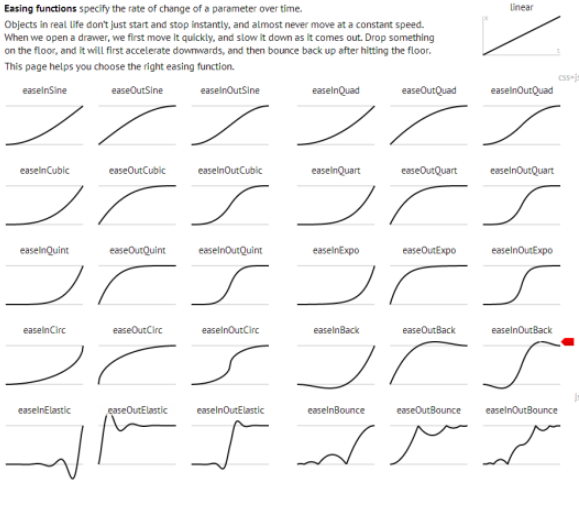
Једна од метода ове библиотеке је приказана на следећи начин:



Ова метода покреће главно тело за коју је закачена (у овом случају transform), до крајње позиције, за 2f (2 pixel-а на хоризонталној оси на сваком ажурирању frame-a), 0f (0 pixel-а на вертикалној оси на сваком ажурирању frame-a), по анимацији која има назив animationCurve. Као додатни аргумент користио сам completeCallback: vracanje аргумент, који као задњи аргумент позива функцију Vracanje(), где је описано да се враћа сам објекат на почетак.



У овој функцији за враћање коју смо позвали, видимо да се сама птичица враћа до почетка, и онда јој се поставља сама позиција на нулу, а када је позиција на нули, креће се испочетка Tween.Position функција.



*Слика 10: Примери кретања помоћу Tween-a*

# Решавање проблема

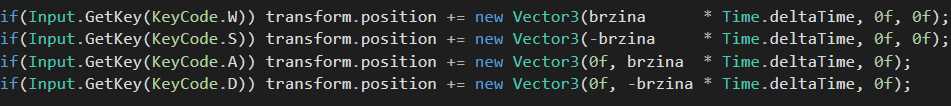
## Кретање објекта

На почетку било је мало теже снаћи се у самом едитору Unity-ја, из разлога што нисам пре програмирао видео игре у неким engine-има. Сам амбијент је постао лакши за разумети и сналажење како је време пролазило, јер само интересовање за програмирање видео игара је било на високом нивоу. Прво сам учио како се стављају објекти на екран, а затим како да правим да се ти објекти крећу на одзив неког тастера на тастатури, тако рецимо поред јако једноставне скрипте:



Ова скрипта помера објекат за одређену брзину сваки фрејм док је притиснут тастер W, и то на x оси.

Наравно може се направити да се она креће путем неког другог вектора, тако рецимо можемо направити кретање у све четири стране помоћу мале надоградње ове скрипте:



Ово ради на јако сличан начин само што овај пут гледа шта је корисник притиснуо и помера саму главну птицу по одређеном вектору за кретање.

Битно је напоменути да овде постоје неколико опција за кретање које Unity подржава. Наиме, постоје Input.GetKey(), Input.GetKeyDown() као и Input.GetKeyUp().

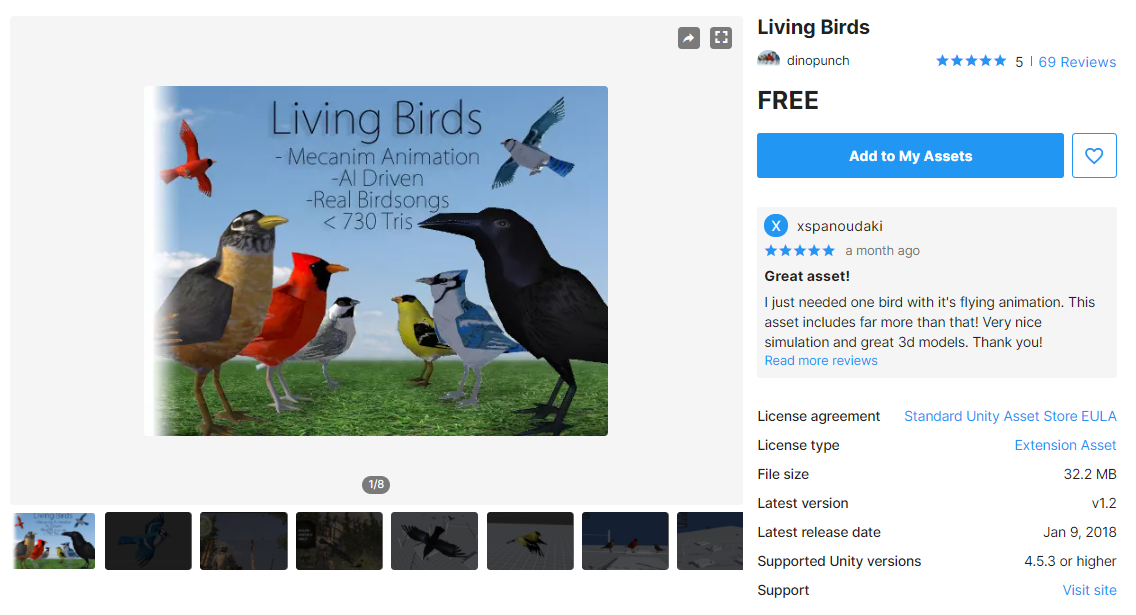
Input.GetKey() је оно што сам ја користио, јер то уствари прелази у TRUE када год се држи неко дугме.

Input.GetKeyDown() реагује само када је притиснут једном неки тастер.

Input.GetKeyUp() реагује када се притиснут тастер пусти.

## Бирање објекта играча и кретање

На почетку мора се изабрати неки објекат за играча који ћемо користити за кретање; то сам одлучио да буде птичица која ће се кретати, летети, скакати и избегавати препреке. Имао сам неколико опција, али када сам видео један пакет птичица, одмах сам одлучио да је то тај. Тај пакет је имао и анимације готове, објекте и боје птичица већ направљен, па сам одмах могао да ступам у програмирање. Као прво, идемо у Asset Store → Search, и након тога тражимо птице.



*Слика 11: Ливинг Бирдс пакет*

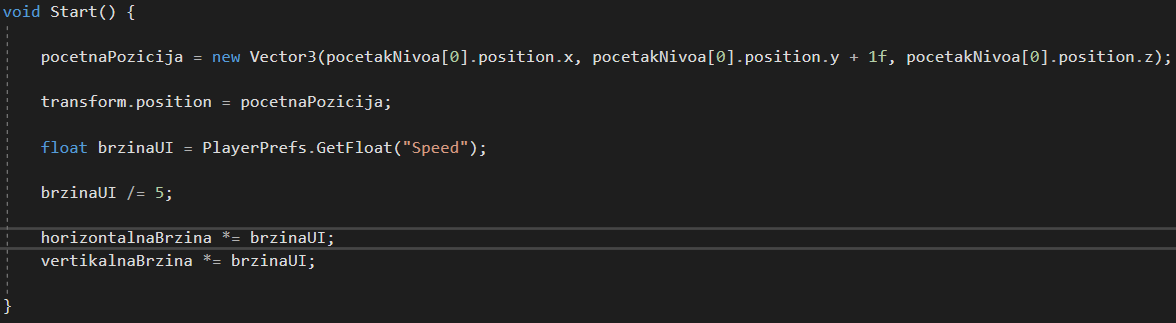
Овде сам одабрао као главног играча црвену птицу „ждрала“, јер је изгледа најпривлачније, и волим црвену боју. Након изабира, поставио сам птицу на претходно направљени први ниво, и након тога, бацио се на писање скрипти за кретање, скакање, као и судар са неким објектом.

Пошто сам знао да ћу имати више нивоа, прво сам декларисао један низ почетних позиција на које ће птица да се ствара у односу на ниво на којем је. Наравно, ниво 0 је почетни ниво.



Након ове декларације, у инспектору сам додао позиције које су у ствари места за стварање, а како видимо, има три нивоа, а самим тим и три почетне позиције.

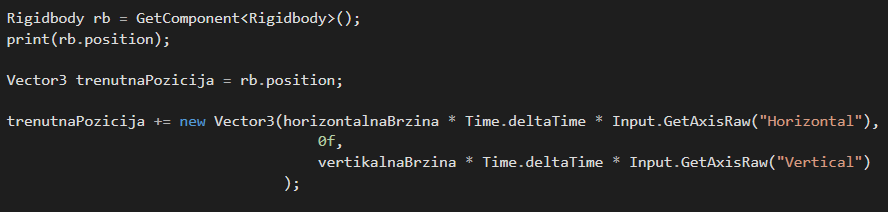
Након овога, написао сам Start() функцију за постављање птице на ту почетну локацију нултог нивоа, а након тога написао јачине брзина хоризонталног и вертикалног кретања.



*Слика 12: Почетна функција у скрипти за кретање*

брзинаУИ представља брзину коју је корисник поставио у подешавањима, више о томе, касније.

Након овога, требало је направити начин за играча да помера птичицу, а то сам урадио на начин како је горе написано било у поглављу 3.1 Кретање играча.



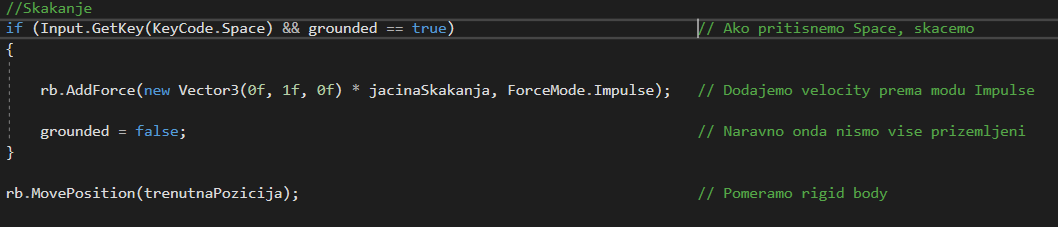
*Слика 13: Кретање играча*

Ово се све дешава у Update() функцији, где видимо да на почетку узимамо RigidBody од саме главне птице, што представља објекат са локацијом објекта за који је закачен. У сваком фрејму узимамо тренутну позицију и онда мануелно са њима радимо. Пошто знамо да је главни циљ птице да се креће само у једном смеру, тако сам и направио функцију која првенствено има циљ да креће птицу напред, а након и наравно гледа ако имамо покретање у страну, ради и то.

Input.GetAxisRaw(“Horizontal”) и Input.GetAxisRaw(“Vertical”), представљају две методе које имају циљ да враћају 0 и 1 као решења. Наиме, ако је притиснуто дугме за кретање по хоризонталној оси, Input.GetAxisRaw(“Horizontal”) враћа вредност 1, што се касније множи са Time.deltaTime и хоризонталном брзином.

## Скакање и летење птице

Скакање је било једно од тежих ствари за рад, из разлога што није једноставно направити птицу да скочи и врати се доле, јер гравитација постоји. Морао сам да експериментишем доста са гравитацијом као и колико да додам птици јачину вектора који је на вертикалној оси, како би скок био најлакши могућ. Ово сам видео да је најлакше урадити по следећем коду:



*Слика 14: Скакање играча*

Овде као и пре, проверавамо да ли је корисник ударио Space, јер је то дугме за скакање; ако није ништа се не дешава. Такође овде је битно да је птица на земљи да може поново да скочи јер ако није, играч може бесконачно да лети и изван мапе и у места где не треба да иде.

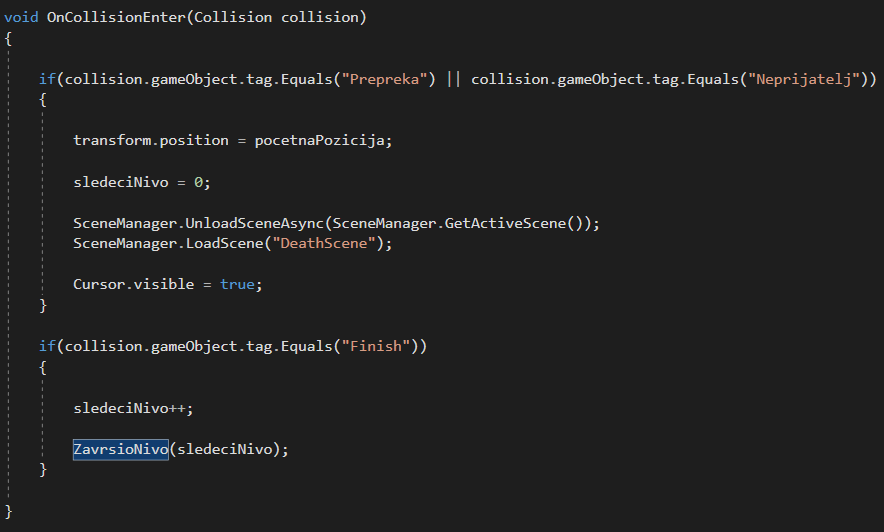
На самом крају функције, померамо само тело на нову тренутну позицију, и тако контролишемо кретање самог објекта.

## Ударац у непријатеља и завршетак нивоа

Ово није било толико тешко за одрадити јер се свело на основни атрибут неког објекта, tag. Таг се користи на то да дамо име или дескрипцију неком објекту у виду неког стринга и онда можемо даље да радимо са тим објектом или објектима. У овој игрици користио сам само три тага, „Непријатељ“, „Препрека“ и „Finish”.

Ако се птица удари у непријатеља или препреку, игрица је готова, није успео да је пређе, и враћа га на почетни мени. Као што видимо сада, главни циљ игре је да не удари ни у шта како би дошао до краја без проблема.

Ако се птица удари у *Finish,* то значи да је завршио ниво и иде даље.



*Слика 15: Ударање птице у неки објекат*

Ако се удари у непријатеља или препреку, враћа се позиција на почетну позицију, следећиНиво се поставља на нулу, и активирамо сцену *DeathScene*, која онда даље има опцију да поново крене игрицу, или да изађе.

Ако се удари у *Finish,* следећи ниво променљива се повећава за један, и позива се функција ZavrsioNivo(int; ), која онда активира сцену следећу за покретање следећег нивоа.

Након свега овога кодирања, имамо птицу која је постављена на почетну позицију и може се кретати напред, назад, лево и десно.



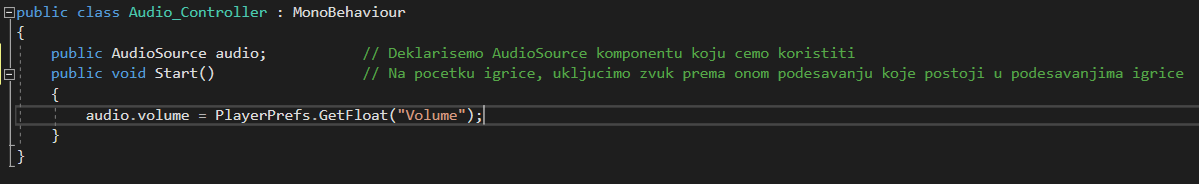
*Слика 16: Изглед почетка видео игре*

Битно је напоменути да се треба мало експериментисати са камером и светлом. У мом примеру, ја сам повезао светло и камеру за главну птицу, тако да како се ми крећемо, крећу се и они за њом и прате је. Неке видео игре можда имају фиксну камеру, или светло, али овде је поента да цело место буде јако тамно, а светло осветљава само мало светла испред себе.

Видимо дрва и ограду, што су све препреке које ако играч дотакне, пропада и ниво му је прекинут као и цела игрица и враћен је на почетак.

## Звук и музика

Као и свака игрица, тако и ова има музику која се покреће када крене први ниво. Као и објекти, може се и музика скинути са Asset Store-а; тако сам ја скинуо један пакет за музику који сам користио за рад са звуком.

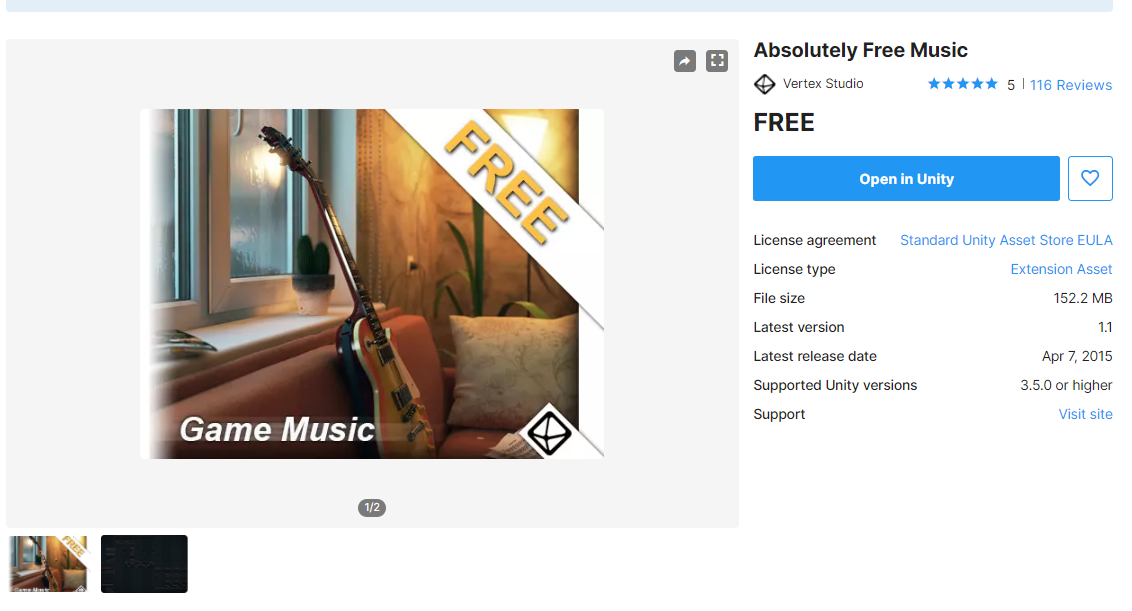


*Слика 17: Аудио контролер за звук*

Декларисали смо један AudioSource на почетку, и у инспектору додали музику у њега.

На почетку видео игре, поставили јачину звука ове музике на одређени степен, како је корисник изабрао у подешавањима.

Назив пакета је *Absolutely Free Music,* од *Vertex* студија.



*Слика 18: Изглед музике у Asset Store-u*

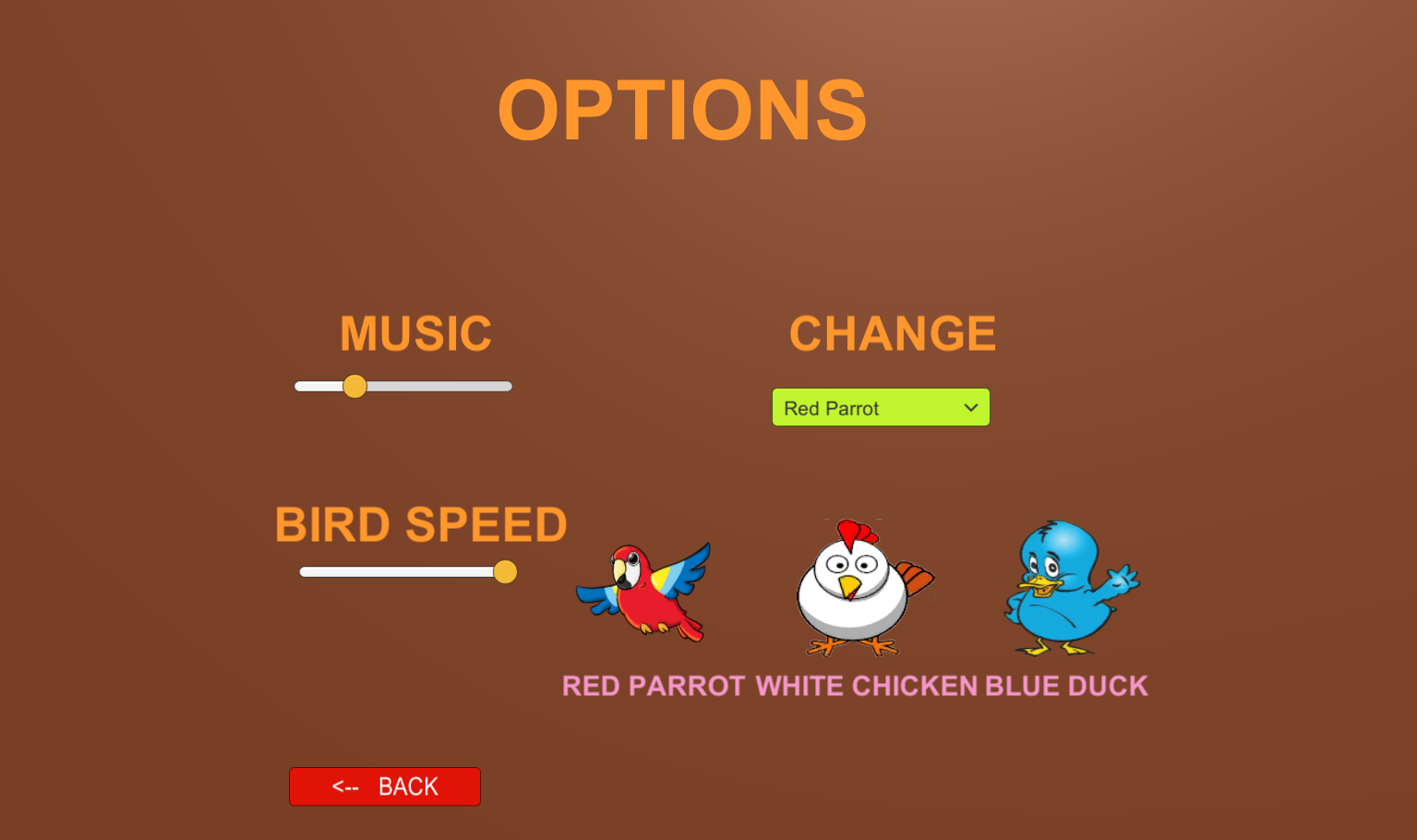
## Опције

Корисник у главном менију видео игре има могућност да уђе у опције, у којима се налази мноштво ствари. Овом приликом он може да промени јачину звука; повећа односно смањи брзину кретања птичице и промени боју птичице. Имају три понуђене боје: бела, плава и црвена, и свака боја утиче на саму птичицу.

Два слајдера су постављена за промену брзине птице и јачину звука, које имају пет статичних вредности.

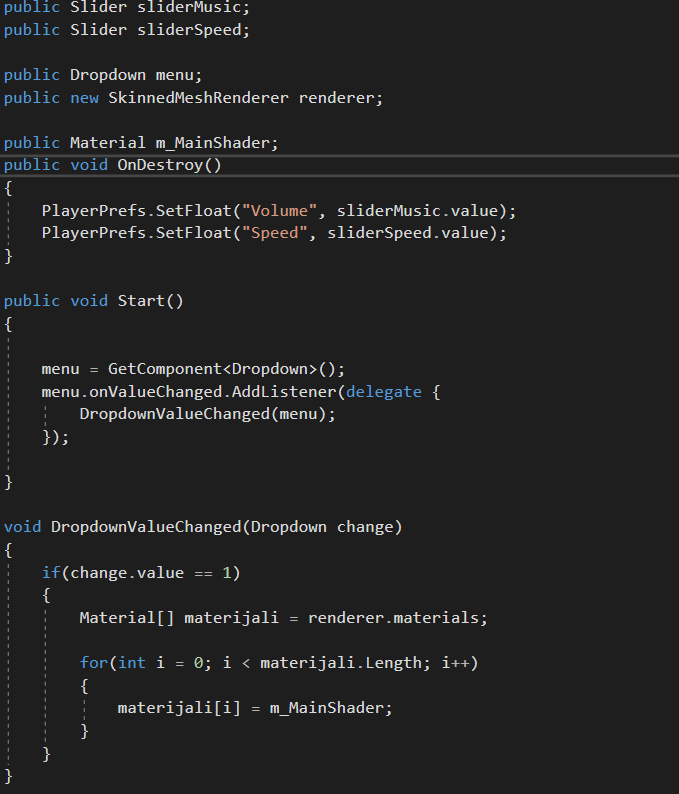
За брзину, она може да се смањи чак пет пута!

За јачину звука, она иде од 0.2 до 1.0, у подељцима од 0.2. На почетку је звук постављен на 0.4, па корисник може да промени јачину звука како жели.



*Слика 19: Опције и подешавања видео игре*

Наравно корисник увек може да се врати назад у главни мени и уђе у неко друго подешавање видео игре, или да изађе из ње или крене нови покушај.



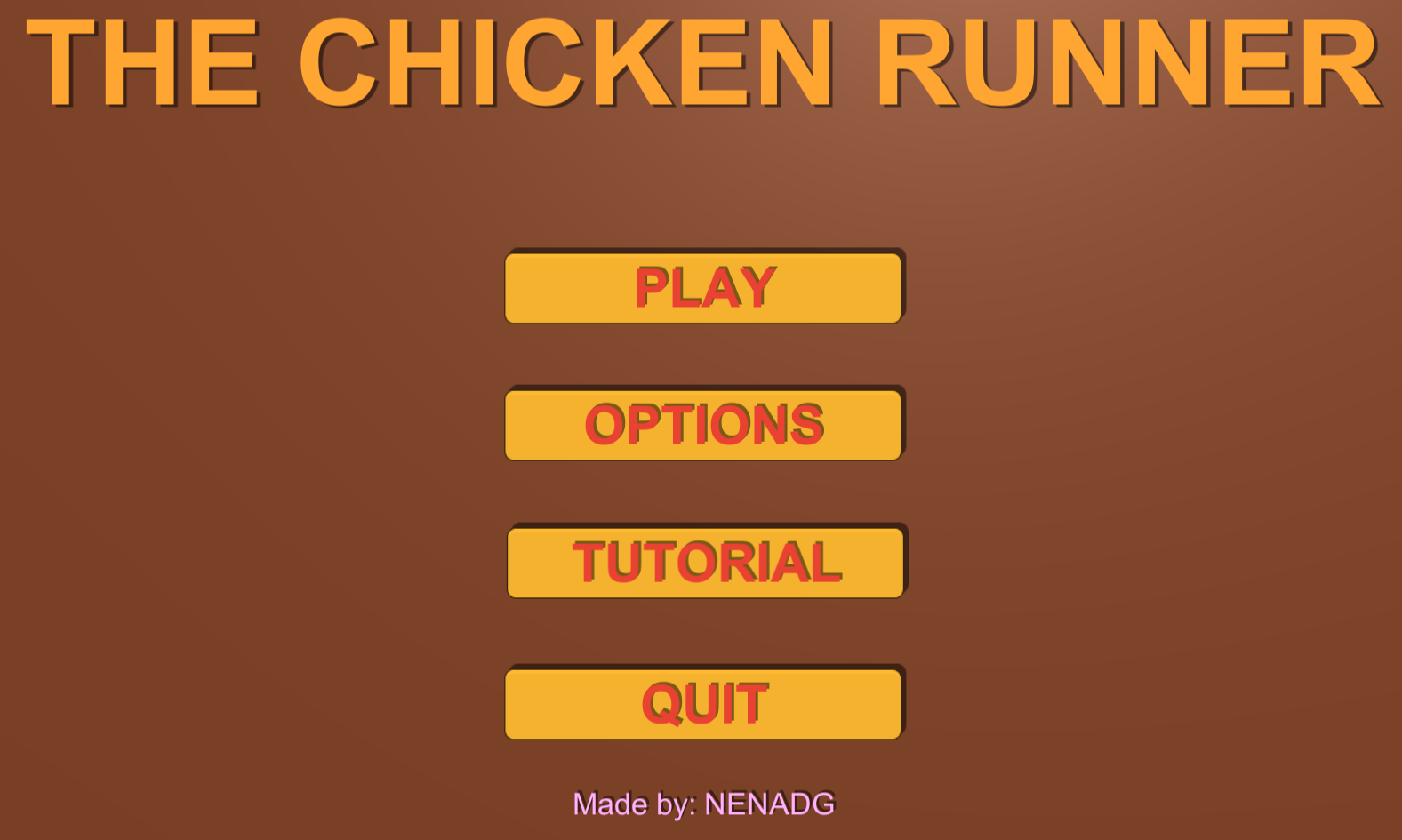
*Слика 20: Код за подешавања видео игре*

У ОnDestroy() методи поставља се вредност вредности *Volume* и *Speed* на одређене вредности слајдера.

Start() функција је у овом случају коришћена да се промени боја птичица при старту видео игре.

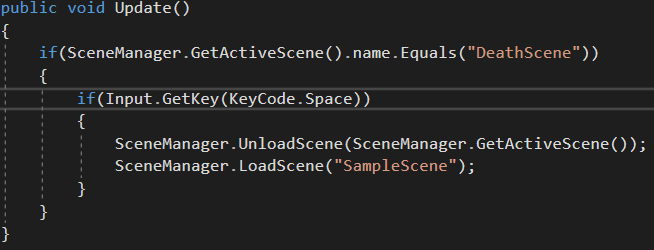
## Главни мени

Главни мени мора да постоји у свакој игрици, па тако постоји и у овој. У главном менију се налазе опције корисника да уђу у нову игру, у туторијал или опције, или да изађу из саме видео игре.

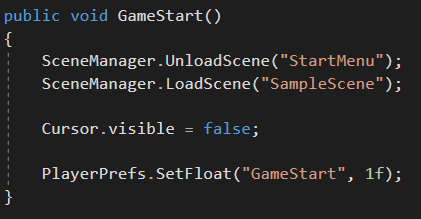


*Слика 20: Главни мени видео игре*

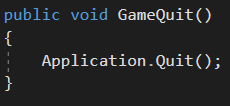
Када корисник стисне *Play,* улази му у нову игру, покреће тајмер, и одбројава време које му је потребно да заврши нивое.



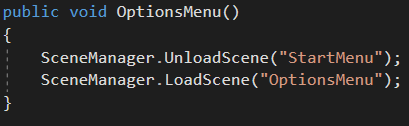
*Слика 21: Приказ активирања почетка видео игре ако корисник стисне space*



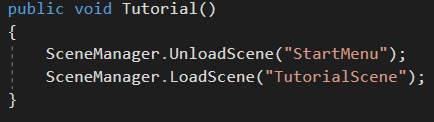
*Слика 22.а: Приказ Start дугмета*



*Слика 22.б: Приказ Quit дугмета*



*Слика 22.в: Приказ Options дугмета*



*Слика 22.г: Приказ Tutorial дугмета*

## Победа

Уколико играч победи и пређе игрицу, битно је обавестити га о томе, па му пуштамо сцену за победу и исписујемо његово време, за које му је требало да заврши игрицу.



*Слика 23: Сцена победе*

Играч овде може да се врати у мени и игра поново, или да изађе комплетно из видео игре ако је задовољан временом.

Ако одлучи да поново игра игрицу, постављамо главног играча на почетну позицију, ниво на нула, и игра поново креће.

# Закључак

Прављење видео игре је било веома занимљиво, као и цела радионица прављења игрица. Смем да кажем да су ментори кроз радионицу сваки пут били веома љубазни и пријатни, и све похвале за њих дајем. Веома сам уживао и жељно ишчекивао сваким даном да дође субота када смо имали радионице. C# језик је веома интересантан, и сличан јави са којом сам радио, и даље радим, па сам се лако снашао у синтакси и логици самог језика. Unity је одлична ствар и едитор за видео игре, јер је јако пријатан за нове кориснике, као и све који улазе у свет програмирања видео игара, а могу рећи и веома је моћан из разлога што можемо свашта са њим радити.

Игрица је компатибилна са свим Десктоп рачунарима, као и MacOS, Android и Apple уређајима.

Игрица је веома добро испала; са њом сам освојио и прво место у тој радионици, па могу рећи да сам био успешан. Ако будем имао времена кроз наредне године, дефинитивно ћу се бацити више у програмирање видео игара, јер ми се јако свиђа поступак и сам рад.

Литература

[1] Информације о Unity платформи, <https://en.wikipedia.org/wiki/Unity_(game_engine)>

[2] Скидање платформе, <https://unity.com/>

[3] Asset Store, <https://assetstore.unity.com/>

[4] Living Birds пакет, <https://assetstore.unity.com/packages/3d/characters/animals/birds/living-birds-15649>

[5] Free Trees пакет, <https://assetstore.unity.com/packages/3d/vegetation/trees/free-trees-103208>

[6] Absolutely Free Music пакет, <https://assetstore.unity.com/packages/audio/music/absolutely-free-music-4883>

[7]Visual Studio, <https://visualstudio.microsoft.com/>

[8] Lauda, <https://lauda-edu.org/>