LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

3D SPĒLE UN TĀS MĀJASLAPA

KVALIFIKĀCIJAS DARBS

Autors: **Pāvels Jacuks**

Studenta apliecības Nr.:pj13005

Darba vadītājs: M.Dat Arturs Lavernovs

Rīga – 2015

ANOTĀCIJA

Kvalifikācijas darbu izstrādāja Pāvels Jacuks kā daļu no pirmā līmeņa profesionālās studiju programmas Programmēšana un datortīklu administrēšana. Kvalifikācijas dara vadītājs ir Artūrs Lavernovs. Kvalifikācijas darba 3D spēles un mājas lapas nosaukums ir „GizmoSPEED".

Spēles un mājas lapas „GizmoSPEED" **raksturojums:** lidojot cauri tunelim, kas ik reizi tiek ģenerēts no jauna, spēlētājam jāizvairās no šķēršļiem un jāsakrāj pēc iespējas vairāk punktu. Kad tiek sasniegts augsts rezultāts, tiek piedāvāta izvēle saglabāt rezultātu *MySQL* datubāzē ar iespēju aplūkot to mājas lapas rezultātu sadaļā. Spēle ir *3D* un tika veidota C# valodā ar *4.0 XNA* un *DirectX* atbalstu. Mājas lapa tika izstrādāta ar *Laravel 4.2.16 un MySQL* atbalstu.

Kvalifikācijas darba **mērķis** ir nostiprināt un iegūt jaunas zināšanas par spēļu programmēšanu, spēļu materiāla izstrādi(*3D modelēšana, animēšana*) un mājas lapu izstrādi. Spēle „GizmoSPEED" ir izstrādāta uz *PC* platformas.

Kvalifikācijas darba **uzdevums** ir spēles „GizmoSpeed" programmēšana un spēles mehāniku izstrāde priekš PC platformas, spēles optimizācija un testēšana. Programmētā koda analizēšana, kas ir iekļauta dokumentācijā.

Atslēgas vārdi: Spēle; C#; XNA 4.0; 3D;

ABSTRACT

Qualification paper was made by Pāvels Jacuks, as a part of study program *Programming and computer network administration.* Qualification paper's manager is Arturs Lavernovs. Qualification paper's title of 3D game and website is „GizmoSPEED".

Description of the game and website „GizmoSPEED": the protagonist has to fly through a tunnel, which is generated randomly every time you play, avoiding every obstacle in order to progress. Each second you gain bigger score while protagonist is unharmed and when you hit an obstacle, you get an option to save your highscore in MySQL database. You can also view the highscore on the website.

Qualification paper's **target** is to straighten and gain knowledge about the game programming, developing game assets (*3D models, animations*), and website development. The game "GizmoSPEED" is developed for PC platform.

Qualification paper's **tasks** are developing the games "GizmoSPEED" mechanics, programming for PC platform, game optimization and testing. Also the programming code is analyzed and posted in this qualification paper.

Keywords: Game; C#; XNA 4.0; 3D;

SATURS

IEVADS

Darba autors ir Pāvels Jacuks. Viens no svarīgākajiem procesiem spēles veidošanā ir programmatūras izstrāde. Iespējams, tas ir pats svarīgākais izstrādes posms, jo uz izveidotā koda balstās visa spēle. Visas izveidotās animācijas, grafika un sintezētās skaņas var būt lieliskas, bet, ja nav izstrādāts nepieciešamais kods, tad visas šīs lietas kļūs neizmantojamas. Spēļu izveide ir sarežģīta tādēļ, ka iekļauj tādus elementus kā programmēšana, mehāniku izstrāde un stāsta veidošana. Bez minētajiem elementiem spēlei vēl ir nepieciešamas specifiskās sastāvdaļas – grafika, skaņa un izvēlnes (*Menu*). Šī kvalifikācijas darba ietvaros prezentēto spēli esmu veidojis pats, bez papildu 3D mākslinieku vai dizaineru palīdzības. Lai pašam izveidotu savas tuneļu, protagonista un šķēršļu 3D modeļus, *rigotu* (pievienot skeletu, lai modeļi varētu kustēties) un *tekstūras*, ir nepieciešama gadiem ilga pieredze. Spēles sastāvdaļu izstrādāšana sasniegtajā kvalitātē ir darbietilpīgs process. Lai to izprastu, piedāvāju aplūkot ekrānuzņēmumus (skat. 65.lpp.).

Spēles un mājas lapas „GizmoSPEED" **raksturojums**: lidojot cauri tunelim, kas ik reizi tiek ģenerēts no jauna, spēlētājam jāizvairās no šķēršļiem un jāsakrāj pēc iespējas vairāk punktu. Kad tiek sasniegts augsts rezultāts, tiek piedāvāta izvēle saglabāt rezultātu *MySQL* datubāzē ar iespēju aplūkot to mājas lapas rezultātu sadaļā. Spēle ir *3D* formātā un tika veidota C# valodā ar *4.0 XNA* un *DirectX* atbalstu. Mājas lapa tika izstrādāta ar *Laravel 4.2.16 un MySQL* atbalstu.

Kvalifikācijas darba **mērķis** ir nostiprināt un iegūt jaunas zināšanas par spēļu programmēšanu, spēļu materiāla izstrādi (*3D modelēšana, animēšana*) un mājas lapas izstrādi. Spēle „GizmoSPEED" ir izstrādāta *PC* platformai.

Kvalifikācijas darba **uzdevums** ir spēles „GizmoSpeed" programmēšana un spēles mehāniku izstrāde PC platformai, spēles optimizācija un testēšana, kā arī programmētā koda analīze, kas ir iekļauta dokumentācijā.

1. Programmatūras prasību specifikācija

1.1 Ievads

1.1.1. Nolūks

Programmatūras prasību specifikācija (turpmāk tekstā –PPS) ir izstrādāta Latvijas Universitātes 2. kursa kvalifikācijas darba ietvaros un ir paredzēta programmprodukta „GizmoSPEED" prasību aprakstīšanai.

Dokumenta nolūks ir precīzi un viennozīmīgi formulēt sistēmas prasības un raksturot tās funkcionalitāti. PPS ir paredzēta izstrādātājiem, kuri veiks programmprodukta projektējuma un paša programmprodukta izstrādi, un pasūtītāja pārstāvjiem, kuri apstiprina PPS.

1.1.2. Darbības sfēra

Programmprodukts „GizmoSPEED" ir spēle, kas ir domāta jautrākai brīvā laika pavadīšanai. Šajā spēlē lietotāji var izmēģināt savas koncentrēšanās spējas un reakcijas ātrumu, kā arī sacensties ar draugiem, salīdzinot savus augstākos rezultātus. Produkts sastāv no divām daļām – spēles un tīmekļa vietnes.

Sistēmas funkcionalitāte ir nodrošināta - lietotājiem ir iespēja lejuplādēt un uzstādīt uz datora spēlei nepieciešamo datni, kas ir pieejama tīmekļa vietnes galvenajā lapā.

1.1.3. Definīcijas un saīsinājumi

PPS – Programmatūras prasību specifikācija.

PPA – Programmatūras projektējuma apraksts.

1.1.4. Saistība ar citiem dokumentiem

Dokumenta noformēšana ievērotas standarta LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis" prasības.

1.1.5. Pārskats

Dokuments sastāv no 4 daļām:

1. **Ievads.** Ievadinformācijas, kas satur dokumenta nolūku un sniedz nepieciešamo termiņu skaidrojumus, kas sastopami šajā dokumentā un norāda saistību ar citiem dokumentiem.
2. **Vispārējs apraksts.** Aprakstīta produkta perspektīva un galvenās tā funkcijas, galvenās lietotāju grupas, kā arī galvenie produkta ierobežojumi.
3. **Funkcionālās prasības.** Detalizēti aprakstīti funkciju mērķi, ievaddati, apstrāde, izvaddati un kļūdu paziņojumi.
4. **Nefunkcionālās prasības.** Apraksta prasības pret sistēmu kopumā – pieejamība, datu drošība u.c.

1.2.1. Produkta perspektīva

Produkts ir paredzēts lietotāju izklaidei, īpaši cilvēkiem, kas ir aizrauti ar spēlēm un multivides izklaides industriju. Programmatūra ir veidota ar tālākas attīstības, uzlabojumu iespējam. Pēc kvalifikācijas darba aizstāvēšanas izmantošu vienu no jaunu spēļu mārketinga taktikām. Tā ir spēles iesūtīšana spēļu recenzentiem un cilvēkiem, kas nodarbojās ar tiešraidi, kuriem jau ir izveidojies savs skatītāju loks. Ja vismaz viena šāda persona piekritīs, tad manai tīmekļa vietnei pieaugs apmeklētāju skaits. Šajā gadījumā es ievietošu reklāmu ar abpusēji izdevīgu Google AdSence sadarbību, kas ļaus man ģenerēt maksu par „GizmoSPEED" mājaslapas apmeklējumiem.

1.2.2. Produkta funkcijas

Sistēmai ir jāspēj nolasīt ievaddatus no lietotāja tastatūras un peles. Sistēmai ir jāpiedāvā iespēju saglabāt savu sasniegto rezultātu, un saglabāt katru iesūtīto rezultātu datubāzē. Sistēmai ir jāļauj sākt spēli no jauna un dot iespēju iziet uz galveno izvēlni (*Menu*). Sistēmai ir jāļauj pielāgoties lietotāja ekrāna izšķirtspējai, kā arī dot iespēju iziet no spēles. Sistēmai jāļauj atstāt komentārus „GizmoSPEED" tīmekļa vietnē.

1.2.3. Lietotāju grupas

Sistēmai ir tikai viena lietotāju grupa – sistēmas lietotājs. Sistēmas lietotājiem ir pieejamas visas iespējamās funkcijas bez ierobežojumiem.

1.2.4. Lietotāju raksturiezīmes

Sistēmas lietotājiem ir jābūt pamatzināšanām darbā ar datoru un interneta pārlūkprogrammu. Galvenā mērķauditorijas raksturiezīme ir interese spēlēt „GizmoSPEED".

1.2.5. Vispārējie ierobežojumi

Lietotājam ir nepieciešams dators ar Windows operētājsitēmu, uz kura ir uzstādīta spēlei nepieciešama datne, kuru var lejuplādēt no „GizmoSPEED" tīmekļa vietnes.

1.3. Funkcionālās prasības

1.3.1. Spēles galvenās izvēlnes funkcijas

1.3.1.1. Spēles uzsākšana

Identifikators: GizmoSPEED.gameStarted
Mērķis
Funkcija ļauj lietotājam uzsākt jaunu spēli, un dod iespēju ģenerēt jaunus tuneļus un šķērš pēc nejaušības principa (*random*).
Ievaddati
Lietotājs veic sekojošas darbības: 1. Ieslēdz spēli. 2. Spiež „Play” pogu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem datus - ja spēle jau bija iepriekš spēlēta, bet vēl netika izslēgta. 1.1. Izmantojot spēles klases un tuneļu klases *reset()* metodi, ģenerē jaunus tuneļus šķēršļu pēc nejaušības principa. 2. Funkcija saņem datus – ja spēle tikko ieslēgta. 2.1. Izmantojot spēles klases un tuneļu klases *loadContent()* metodi, ģenerē jaun tuneļus un šķēršļu pēc nejaušības principa.
Izvaddati
Ir sagatavots spēles laukums.
Kļūdu paziņojumi
Nav
Komentāri
Spēles funkcionalitāte lietotājiem ir pieejama, lejuplādējot un uzstādot uz datora spēlei nepieciešamo datni, kas ir pieejama tīmekļa vietnes galvenajā lapā.

1.3.1.2 Iziešana no spēles.

Identifikators: GizmoSPEED.Exit()
Mērķis
Funkcija ļauj lietotājam iziet no spēles.
Ievaddati
Lietotājs veic sekojošas darbības: 1. Spiež „Exit” pogu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem datus par spēles beigšanu. 1.1. Spēle atbrīvo datora atmiņu no aizņemtās informācijas, tad pārtrauc un izslēdz spēles klientu.
Izvaddati
Tiek pārtraukta un spēles klienta darbīga un spēle tiek izslēgta.
Kļūdu paziņojumi
Nav
Komentāri
No spēles var iziet tikai no galvenās izvēlnes.

1.3.1.3 Iestatījumu atvēršana (*opcijas*)

Identifikators: GizmoSPEED.UI.showOptions
Mērķis
Funkcija ļauj lietotājam atvērt opcijas, kurās ir pieejama iespēja mainīt spēles izšķirtspēju un uzlabot spēles darbību uz sava datora.
Ievaddati
Lietotājs veic sekojošas darbības: 1. Spiež „Options” pogu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem datus – ja opcijas logs vēl nav atvērts. 1.1. Tiek atvērts jauns logs ar opcijām. 2. Funkcija saņem datus – ja opcijas logs jau ir atvērts. 2.1. Funkcija atvērt opcijas neizpildās.
Izvaddati
Tiek atvērts jauns logs ar opcijām.
Kļūdu paziņojumi
Nav
Komentāri
Opcijās pieejama iespēja mainīt spēles izšķirtspēju un uzlabot spēles darbību uz sava datora.

1.3.1.4 Spēles ekrāna izšķirtspējas mainīšana iestatījumos

Identifikators: GizmoSPEED.UI.resolution[,]
Mērķis
Funkcija ļauj lietotājam mainīt spēles izšķirtspēju (*resolution*).
Ievaddati
Lietotājs veic sekojošas darbības: 1. Spiež „Options” pogu. 2. Uzstāda sev piemēroto izšķirtspēju. 3. Spiež „Apply” pogu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem datus – ja lietotāja ekrāns atbalsta izvēlēto izšķirtspēju. 1.1. Spēles klienta izšķirtspēja tiek mainīta uz izvēlēto. 2. Funkcija saņem datus – ja lietotāja ekrāns neatbalsta izvēlēto izšķirtspēju. 2.1. Funkcija *izmet* kļūdas paziņojumu 1.
Izvaddati
Spēles klienta izšķirtspēja tiek mainīta uz izvēlēto.
Kļūdu paziņojumi
1. Ekrāns neatbalsta izvēlēto izšķirtspēju.
Komentāri
Noklusētais izšķirtspējas iestatījums ir 1336x768 (16:9). Ir pieejamas visas iespējamās izšķirtspējas: (3:4) 800x600; 1152x864; 1400x1050; 1600x1200; (16:9) 1024x600; 1280x720; 1366x768; 1680x1050; 1920x1080; Spēles izšķirtspēja var ietekmēt spēles darbību – ar mazāku izšķirtspēju spēle darbosies ātrāk datoriem bez labas videokartes un procesora.

1.3.1.5 Spēles saskarnes pozīciju atjaunošana pēc izšķirtspējas maiņas

Identifikators: GizmoSPEED.UI.Button.UpdatePosition()
Mērķis
Kad lietotājs pamaina spēles izšķirtspēju, visas pogas un foni tiek zīmēti vietās, kur tie b zīmēti pirms izšķirtspējas maiņas. Šī funkcija to novērš.
Ievaddati
Lietotājs veic sekojošas darbības: 1. Spiež „Options” pogu. 2. Uzstāda sev piemēroto izšķirtspēju. 3. Spiež „Apply” pogu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem datus – ja lietotāja ekrāns atbalsta izvēlēto izšķirtspēju. 1.1. Katram saskarnes elementam (pogām, zīmējumiem) tiek izsaukta meto *UpdatePosition(),* kas rediģē to atrašanas vietu uz jaunu pēc izšķirtspējas maiņas. 2. Funkcija saņem datus – ja lietotāja ekrāns neatbalsta izvēlēto izšķirtspēju. 2.1. Funkcija *izmet* jaunu logu ar kļūdas paziņojumu.
Izvaddati
Pēc spēles klienta izšķirtspējas maiņas visu saskarnes elementu atrašanās vietas tiek rediģētas uz jaunām.
Kļūdu paziņojumi
Ja lietotāja ekrāns neatbalsta izvēlēto izšķirtspēju, funkcija *izmet* jaunu logu ar kļūd paziņojumu.
Komentāri
Nav.

1.3.1.6 Spēļu animāciju izslēgšana/ieslēgšana iestatījumos.

Identifikators: GizmoSPEED.UI.animation
Mērķis
Funkcija ļauj lietotājam atslēgt spēles animācijas ar nolūku uzlabot spēles darbību.
Ievaddati
Lietotājs veic sekojošas darbības: 1. Spiež „Options” pogu 2. Uzstāda sev animācijas opciju („ON” vai „OFF”). 3. Spiež „Apply” pogu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem datus – ja lietotājs izvēlējās atslēgt animāciju. 1.1. Katram 3D elementam, kas ir animēts, tiek atslēgta animācija. 2. Funkcija saņem datus – ja lietotājs izvēlējas ieslēgt animāciju. 2.1. Katram 3D elementam, kas ir animēts, tiek ieslēgta animācija.
Izvaddati
Pēc spēles opcijas apstiprinājuma, katram 3D elementam, kas ir animēts, tiek atslēgta ieslēgta animācija, atkarībā no uzstādītās opcijas.
Kļūdu paziņojumi
Nav.
Komentāri
Noklusētais animācijas iestatījums ir ieslēgts. Izslēdzot to, var uzlabot spēles darbību datoriem bez jaudīga procesora. Šī opcija neattiecas uz videokarti.

1.3.1.7 Iziešana no iestatījumiem.

Identifikators: GizmoSPEED.UI.showOptions
Mērķis
Kad iestatījumu logs ir atvērts, funkcija ļauj lietotājam no tā iziet.
Ievaddati
Lietotājs veic sekojošas darbības: 1. Spiež „Back” pogu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem datus – ja lietotājs izvēlējās iziet no iestatījumu loga 1.1. Iestatījumu logs tiek aizvērts.
Izvaddati
Iestatījumu logs tiek aizvērts.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2 Spēles funkcijas.

1.3.2.1 Spēles galvenā personāža lidošanas virziena kontrole.

Identifikators: GizmoSPEED.playerRotation;
Mērķis
Spēlē jābūt iespējai mainīt galvenā personāža lidošanas virzienu, lai lietotājs varētu izvairīt no šķēršļiem.
Ievaddati
Lietotājs veic sekojošas darbības: 1. Izmantojot peli, manevrē galveno personāžu starp šķēršļiem.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem lietotāja patreizējās un iepriekšējās peles pozīcijas datus. 1.1. Izmantojot iepriekšējās un patreizējās peles pozīcijas deltu, tiek aprēķināts galvenā personāža skata virziens.
Izvaddati
Tiek manīts galvenā personāža virziens.
Kļūdu paziņojumi
Nav.
Komentāri
Spēli ir vieglāk spēlēt bez straujām peles kustībām.

1.3.2.2 Spēles galvenā personāža lidošanas pozīcija atkarībā no virziena.

Identifikators: GizmoSPEED.playerPos (Vector3)
Mērķis
Spēlē jābūt iespējai atpazīt galvenā personāža virzienu (*rotāciju*) un, ņemot to vērā, rediģēt pozīciju, lai tas lidotu pareizajā virzienā.
Ievaddati
Lietotājs veic sekojošas darbības: 1. Izmantojot peli, manevrē galveno personāžu starp šķēršļiem.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem galvenā personāža rotācijas datus. 1.1. Tiek aprēķināts, cik jāpieskaita pie galvenā personāža pozīcijas vektora (*X, Y, Z*), izmantojot rotācijas sinusu un kosinusu aprēķinus.
Izvaddati
Izrēķinātās mērvienības tiek pieskaitītas pie galvenā personāža pozīcijas vektora tā, lai kustība uz priekšu norit galvenā personāža virzienā.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2.4 Spēles galvenā personāža lidošanas ātruma aritmētiskā progresija.

Identifikators: GizmoSPEED.playerSpeed
Mērķis
Jo ātrāks ātrums, jo grūtāk izvairīties no šķēršļiem, tāpēc , lai spēle būtu interesanta, tika pievienota galvenā personāža lidošanas ātruma pakāpeniska izaugsme.
Ievaddati
Lietotājs veic sekojošas darbības: 1. Izmantojot peli, manevrē galveno personāžu, neietriecoties šķēršļos.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem galvenā personāža ātruma datus. 1.1. Pēc katras nolidotās spēles vienības, ātrumam tiek pieskaitīts 0.03 vienības.
Izvaddati
Ātrums paliek ātrāks pēc katras nolidotās spēles vienības.
Kļūdu paziņojumi
Nav.
Komentāri
Ātrumam nav robežu.

1.3.2.5 3D kameras pozīcija spēlē.

Identifikators: GizmoSPEED.Camera
Mērķis
Pilnīgi katrai 3D spēlei ir vajadzīga 3D kamera, pat ja tā ir statiska - bez tās spēles dzinis nezinās, kāds ir dots projicējums, kādā izšķirtspējā un kur attiecība pret ekrānu ir jāzīmē 3D attēls.
Ievaddati
Sekojošas darbības: 1. Izmanto galvenā personāža pašreizējo pozīciju.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem galvenā personāža pašreizējās pozīcijas datus. 1.1. Pēc katras nolidotās spēles vienības, 3D kamera precīzi pielāgojās galvenā personāža pozīcijai.
Izvaddati
3D kamera pielāgojās galvenā personāža pozīcijai.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2.6 3D kameras rotācija spēlē.

Identifikators: GizmoSPEED.Camera
Mērķis
Sasniegt gludu kameras virziena maiņu. Kad lietotājs kustina peli, galvenā personāža orientācija tiek mainīta. Jāpanāk, lai kameras virziens attiecībā pret spēlētāju mainās vienmērīgi (*smooth*), nevis paliek identisks personāža orientācijai. Viens no šīs funkcija mērķiem ir padarīt spēli vizuāli pievilcīgāku.
Ievaddati
Sekojošas darbības: 1. Izmanto galvenā personāža pašreizējo orientācijas virzienu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem galvenā personāža pašreizējās orientācijas datus. 1.1. Pēc katras nolidotās spēles vienības, 3D kamera pakāpeniski pielāgojas galvenā personāža orientācijai.
Izvaddati
3D kamera pakāpeniski pielāgojās galvenā personāža orientācijai.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2.7 3D kameras attāluma palielināšana no galvenā personāža

Identifikators: GizmoSPEED.Camera
Mērķis
Šī funkcija ļauj spēlētājam palielināt attālumu starp 3D kameru un galveno personāžu.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs uz tastatūras spiež un tur pogu „z”.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem attālumu starp personāžu un 3D kameru – ja attālums nepārsniedz atļaujamo attālumu: 1.1. Attālums starp personāžu un 3D kameru palielinās. 2. Funkcija saņem attālumu starp personāžu un 3D kameru – ja attālums pārsniedz atļaujamo attālumu: 2.1. Attālums starp personāžu un 3d kameru paliek nemainīgs.
Izvaddati
Attālums starp personāžu un 3D kameru palielinās.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2.8 3D kameras attāluma samazināšana no galvenā personāža

Identifikators: GizmoSPEED.Camera
Mērķis
Šī funkcija ļauj spēlētājam samazināt attālumu starp 3D kameru un galveno personāžu.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs uz tastatūras spiež un tur pogu „x”.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem attālumu starp personāžu un 3D kameru – ja attālums nepārsniedz pieļaujamo attālumu: 1.1. Attālums starp personāžu un 3D kameru samazinās. 2. Funkcija saņem attālumu starp personāžu un 3D kameru – ja attālums pārsniedz pieļaujamo attālumu: 2.1. Attālums paliek nemainīgs starp personāžu un 3d kameru.
Izvaddati
Attālums starp personāžu un 3D kameru samazinās.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2.9 3D kameras attāluma attālināšana no galvenā personāža

Identifikators: GizmoSPEED.Camera
Mērķis
Šī funkcija ļauj spēlētājam samazināt attālumu starp 3D kameru un galveno personāžu.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs uz tastatūras spiež un tur pogu „x”.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem attālumu starp personāžu un 3D kameru – ja attālums nepārsniedz atļaujamo attālumu: 1.1. Attālums starp personāžu un 3D kameru palielinās. 2. Funkcija saņem attālumu starp personāžu un 3D kameru – ja attālums pārsniedz atļaujamo attālumu: 2.1. Attālums paliek nemainīgs starp personāžu un 3d kameru.
Izvaddati
Attālums starp personāžu un 3D kameru palielinās.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2.10 Jaunu tuneļu ģenerēšana.

Identifikators: GizmoSPEED.TunnelGenerator
Mērķis
Šī funkcija ļauj izveidot jaunu tuneļu sistēmu pēc nejaušības principa.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1.1. Lietotājs ieslēdz spēli. VAI 1.2 . Lietotājs izvēlās atkal spēlēt spēli no jauna. VAI 1.3. Lietotājs izvēlās iziet uz galveno izvēlni.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem atļauju ģenerēt jaunu tuneļu sistēmu. 1.1. Visi mainīgie, tādi kā personāža pozīcija, ātrums, sakrātie punkti u.c., ir atjaunoti uz sākotnējām vērtībām. Tiek ģenerēti jauni, unikāli tuneļi pēc nejaušības principa, līdz ir sasniegts to pieļaujamais skaits.
Izvaddati
Visi mainīgie ir atjaunoti uz sākotnējām vērtībām. Tiek ģenerēti jauni, unikāli tuneļi pēc nejaušības principa.
Kļūdu paziņojumi
Nav.
Komentāri
Tuneļi tiek ģenerēti vienlaicīgi optimizācijas nolūkos. Katram tunelim piemīt sava pozīcija. Spēlē tiks iekļauts **Slepenais tunelis.** Tas tiek ģenerēts ļoti reti, iespēja sastapties ar tādu ir 1/47. Dokumentācijā ir pieejams slepenā tuneļa ekrānuzņēmums un to var aplūkot 68.lpp. *7.3 attēls*

1.3.2.11 Jaunu šķēršļu ģenerēšana.

Identifikators: GizmoSPEED.TunnelGenerator
Mērķis
Šī funkcija ļauj izveidot jaunu šķēršļu sistēmu pēc nejaušības principa.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1.1. Lietotājs ieslēdz spēli. VAI 1.2. Lietotājs izvēlās atkal spēlēt spēli no jauna. VAI 1.3. Lietotājs izvēlās iziet uz galveno izvēlni.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem atļauju ģenerēt jaunu šķēršļu sistēmu. 1.1. Tiek ģenerēti jauni, unikāli šķēršļi pēc nejaušības principa, tie tiek ģenerēti līdz sasniegts to atļaujamais skaits.
Izvaddati
Tiek ģenerēti jauni, unikāli šķēršļi pēc nejaušības principa.
Kļūdu paziņojumi
Nav.
Komentāri
Spēlētājs sastopas ar 9 atšķirīgiem šķēršļiem.

1.3.2.12 Nevajadzīgo tuneļu dzēšana

Identifikators: GizmoSPEED.TunnelGenerator.Tunnel.Drawable
Mērķis
Šī funkcija ļauj atbrīvoties no tuneļu aizņemtās atmiņas. Runa ir par tādiem tuneļiem, kuriem personāžs ir jau palidojis garām. Šīs funkcijas mērķis ir spēles optimizācija.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Personāžs ielido nākamajā tunelī.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem personāža pozīcijas datus. 1.1 Funkcija noskaidro, vai personāža pozīcija atrodas nākamajā tunelī.
Izvaddati
Visi tuneļi, kuriem personāžs palidojis garām, tiek dzēsti.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2.13 Nevajadzīgo šķēršļu dzēšana

Identifikators: GizmoSPEED.TunnelGenerator.Obsticle.Drawable
Mērķis
Šī funkcija ļauj atbrīvoties no šķēršļu aizņemtās atmiņas, kuri ir jau personāžam aiz muguras. Arī šīs funkcijas nolūks ir optimizēt spēli.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Personāžs palido garām šķērslim.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem personāža pozīcijas datus. 1.1 Funkcija noskaidro, vai personāža pozīcija atrodas aiz šķēršļa.
Izvaddati
Visi šķēršļi, kuriem personāžs palidojis garām, tiek dzēsti.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.2.14 Personāža saskarsme ar tuneļa sienām (*Collison Detection*)

Identifikators: GizmoSPEED.TunnelGenerator.Tunnel.CollisionDetection()
Mērķis
Šī funkcija ļauj uzzināt – vai personāžs ir saskāries ar tuneļa sienām, kā rezultātā lietotājs zaudē spēli.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Personāžs ielido, saskaras ar tuneli.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem personāža pozīcijas datus un redzamā tuneļa saskarsmes kasti(*BoundingBox*). 1.1 Funkcija pārbauda, vai personāža pozīcijas punkts saskaras ar tuneļu sienu saskarsmes kastēm (*BoundingBox*).
Izvaddati
Tiek dota spēles pārtraukšanas atļauja, rezultātā parādās nāves animācija, tad uz ekrāna tiek izvadīts izvēlnes logs ar iespējām atjaunot spēli, iziet uz galveno izvēlni vai iesūtīt nopelnīto punktu skaitu.
Kļūdu paziņojumi
Nav.
Komentāri
Paša personāža saskarsmes laukums nav liels. Ja personāžs ar roku aizskars šķērsli vai tun sienu, tad spēle vēl netiek pārtraukta.

1.3.2.15 Personāža saskarsme ar šķēršļiem (*Collison Detection*)

Identifikators: GizmoSPEED.TunnelGenerator.Obstacle.CollisionDetection()
Mērķis
Šī funkcija ļauj uzzināt – vai personāžs ir saskāries ar šķēršļiem, kā rezultātā lietotājs zaudē spēli.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Personāžs ielido, saskaras ar šķēršli.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem personāža pozīcijas datus un redzamo šķēršļu saskarsmes kastēm(*BoundingBox*). 1.1 Funkcija pārbauda, vai personāža pozīcijas punkts saskaras ar šķēršļu saskarsmes kastēm (*BoundingBox*). Rezultātā tiek padota atļauja par spēles pārtraukšanu.
Izvaddati
Tiek dota spēles pārtraukšanas atļauja, rezultātā parādās nāves animācija, tad uz ekrāna tiek izvadīts izvēlnes logs ar iespējām atjaunot spēli, iziet uz galveno izvēlni vai iesūtīt nopelnīto punktu skaitu.
Kļūdu paziņojumi
Nav.
Komentāri
Paša personāža saskarsmes laukums nav liels. Ja personāžs ar roku aizskars šķērsli vai tun sienu, tad spēle vēl netiek pārtraukta.

1.3.2.16 Savāktā rezultāta sūtīšana serverim.

Identifikators: GizmoSPEED.UI.submitScore()
Mērķis
Šī funkcija ļauj sūtīt serverim sakrāto punktu daudzumu. Sūtīšana ir aizsargāta. (skat. apstrāde)
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Izspēlē spēli, līdz ietriecas kādā no šķēršļiem. 2. Redzot iegūto punktu skaitu, spiež pogu „Submit”, tad parādās jauns logs, kur jāieraksta savs vārds. 3. Ievada savu vārdu. 4. Spiež pogu „Submit”.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem sakrāto punktu daudzumu. 1.1 Kad lietotājs raksta savu vārdu, katram nospiestam taustiņam funkcija pārbauda, vai ierakstītais simbols ir derīgs. Nav vajadzīgs kļūdu paziņojums, jo nederīgos simbolus nemaz nevar ievadīt. 1.2 Aizsardzības nolūkos – tiek ģenerēts *Hash string* ar slepenu paroli. Sakrāto punktu daudzums, lietotāja vārds un slepenā parole tiek pārvērsti vienā rindā. No šīs rindas tiek ģenerēts *Hash string.*
Izvaddati
Sakrāto punktu daudzums, lietotāja vārds un *Hash string* tiek aizsūtīts uz servera PHP lapu.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.3. Spēles skaņa

1.3.3.1 Pogas klikšķa skaņas atskaņošana

Identifikators: GizmoSPEED.TunnelGenerator.UI.Button
Mērķis
Šī funkcija nodrošina skaņu, kad lietotājs uzklikšķina uz pogas.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs uzspiež jebkuru pogu.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem lietotāja peles stāvokli. 1.1 Funkcija pārbauda, vai lietotājs ir nospiedis uz pogas.
Izvaddati
Tiek atskaņota klikšķa skaņa.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.3.2 Galvenās izvēlnes fona skaņas atskaņošana

Identifikators: GizmoSPEED.UI
Mērķis
Šī funkcija nodrošina, lai skanētu fona mūzika, kamēr lietotājs atrodas galvenajā izvēlne.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs uzsāk spēli, vai ieiet galvenajā izvēlnē pēc beigtas spēles.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem spēles stāvokli. 1.1 Funkcija pārbauda, vai lietotājs patreiz atrodas galvenajā izvēlnē.
Izvaddati
Tiek atskaņota galvenās izvēlnes fona mūzika.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.3.3 Spēles fona skaņas atskaņošana

Identifikators: GizmoSPEED.TunnelGenerator.tunnelBasicSound
Mērķis
Šī funkcija nodrošina fona mūziku spēles gaitā.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs uzsāk spēli, spiežot pogu „Play”.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem spēles stāvokli. 1.1 Funkcija pārbauda, vai lietotājs patreiz atrodas spēle. Ar katru nolidoto vienību, fona tonalitāte (*pitch*) uzaug.
Izvaddati
Tiek atskaņota fona mūzika.
Kļūdu paziņojumi
Nav.
Komentāri
Ar katru nolidoto vienību fona tonalitāte (*pitch*) uzaug. Tas notiek sinhroni ar spēlēt paātrinājumu, rezultātā skaņa atbilst personāža ātrumam.

1.3.3.4 Šķēršļu skaņas atskaņošana

Identifikators: GizmoSPEED.TunnelGenerator.obstacleSound
Mērķis
Šī funkcija nodrošina skaņu brīdī, kad personāžs palido garām šķērslim.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs lido garām šķērslim.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem personāža un redzamā šķēršļa atrašanas stāvokli. 1.1 Funkcija pārbauda, vai lietotājs ir palidojis garām šķērslim. Tā kā katram šķērslim ir sava skaņa, funkcija pārbauda arī, kādam šķērslim tika palidots garām.
Izvaddati
Tiek atskaņota konkrētā šķēršļa skaņa.
Kļūdu paziņojumi
Nav.
Komentāri
Ar katru nolidoto vienību fona tonalitāte (*pitch*) uzaug. Tas notiek sinhroni ar spēlēt paātrinājumu, rezultātā skaņa atbilst personāža ātrumam. Katram šķērslim jābūt savai skaņai.

1.3.3.5 Personāža nāves skaņas atskaņošana.

Identifikators: GizmoSPEED.TunnelGenerator.obstacleSound
Mērķis
Šī funkcija nodrošina skaņu brīdī, kad personāžs zaudē (ietriecās tuneļa sienā vai šķērslī).
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs saskaras vai ielido tuneļa sienā vai šķērslī.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem atļauju pārtraukt spēli.
Izvaddati
Tiek atskaņota nāves skaņa.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.4. Tīmekļa vietnes funkcijas

1.3.4.1 Atsauksmju atstāšana

Identifikators: GizmoSPEED.submitComment
Mērķis
Šī funkcija nodrošina komentāru atstāšanu „GizmoSPEED" tīmekļa vietnes galvenajā lapā.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs galvenās lapas komentāra lietotājvārda lodziņā ievada savu lietotājvārdu. 2. Lietotājs galvenās lapas komentāra lodziņā ievada savu atsauksmi.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem ierakstīto lietotāja vārdu un atsauksmi. 1.1 Funkcija pārbauda, vai lietotājs ir ievadījis lietotāj vārdu un atsauksmi bez nederīgiem simboliem (nederīgo simbolu gadījumā tiek *izmesta* kļūda 1). 1.2 Funkcija pārbauda, vai lietotāj vārds nepārsniedz atļauto simbolu skaitu (pārsniegšanas gadījumā tiek *izmesta* kļūda 2).
Izvaddati
Atsauksmes dati tiek iesūtīti datu bāzē, tad parādīti uz galvenās lapas.
Kļūdu paziņojumi
1. Your comment contains unsupported symbols. 2. Your nickname is too long.
Komentāri
Nav.

1.3.4.2 Spēles lejupielāde

Identifikators: GizmoSPEED.submitComment
Mērķis
Šī funkcija nodrošina spēles lejupielādi no „GizmoSPEED” tīmekļa vietnes galvenās lapas.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Lietotājs klikšķina lielu, zaļu pogu „Download”.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem pieprasījumu lejupielādēt spēli.
Izvaddati
Internet pārlūkprogramma sāk lejupielādēt datni „GizmoSPEED.rar”.
Kļūdu paziņojumi
Nav.
Komentāri
Nav.

1.3.4.3 Spēles rezultātu iesūtīšana datubāze

Identifikators: GizmoSPEED.score
Mērķis
Šī funkcija nodrošina spēles gaitā savākto punktu daudzumu un lietotājvārda **aizsargātu** iesūtīšanu datubāzē.
Ievaddati
Lietotājs izpilda sekojošas darbības: 1. Izspēlē spēli, līdz ietriecas kādā no šķēršļiem. 2. Redzot iegūto punktu skaitu, spiež pogu „Submit”, tad parādās jauns logs, kur jāieraksta savs vārds. 3. Ievada savu vārdu. 4. Spiež pogu „Submit”.
Apstrāde
Funkcijas apstrāde norit šādi: 1. Funkcija saņem *POST* pieprasījumu, kurā atrodas rinda ar rezultātu. (Saņemtajā rindā glabā tādu informāciju: 1. lietotājvārds + 2. sakrāto punktu skaits + 3. ***Hash string***. ***Hash string*** satur kodētu rindu „lietotājvārds + sakrāto punktu skaits + slepenā parole”) 2. PHP vidē izveido jaunu *Hash string* no lietotājvārda, sakrāto punktu skaita un slepenās paroles. Slepenā parole uz servera ir identiska tai, kas atrodas spēlē. 3. Pārbauda PHP vidē izveidoto *Hash string* ar iesūtīto *Hash string.* *(abam Hash string jāsatur kodēta rinda „lietotājvārds + rezultāts + slepenā parole)* 3.1 Ja iesūtītā Hash string rinda neatbilst PHP vidē izveidotai, tad tiek izmests kļūdas paziņojums 1.
Izvaddati
Datubāzē tiek pievienots jauns rezultāts.
Kļūdu paziņojumi
1. Hash string doesn’t match. Please contact administrator.
Komentāri
Aizsardzības nolūks ir novērst *Packet sniffer* datu manipulēšanu.

1.4. Nefunkcionālas prasības

1.4.1. Lietotāja saskarsme

Sistēma realizēta, galvenokārt izmantojot grafisko saskarni, kā arī visas pogas un darbības ir angļu valodā.

Lietojumprogrammai jāspēj pielāgoties dažādām izšķirtspējām un dažādu lielumu ekrāniem.

1.4.2. Veiktspējas prasības

Tīmekļa vietnes lietošanai jāpietiek tikai ar tīmekļa pārlūkprogrammu, savukārt „GizmoSPEED" lietotājiem spēles funkcionalitāte ir pieejama, lejupielādējot un uzstādot uz datora spēlei nepieciešamo datni. Uz datora ar Intel Core i7-3630QM(2.4GHz) un pietiekami vidējo videokarti, spēlei ir jānorit vienmērīgi.

1.4.3. Atmiņas patēriņa prasības

Spēlējot spēli vienu vai vairākas reizes, sistēma nedrīkst patērēt vairāk par 2GB RAM. Spēles laikā vidēji tiek aizņemti 570-640 MB operatīvās atmiņas.

1.4.4. Drošības prasības

- Sistēmai jānodrošina, lai parasts lietotājs nevar izdzēst uz servera esošus datus.
- Sistēmai jānodrošina, lai lietotājs nevar rediģēt, noķert un rediģēt izsūtītos datus no spēles uz serveri (izmantojot *Packet sniffers*, piemēram, lietojumprogrammatūra *WPE Pro*).

2. Programmatūras projektējuma apraksts

2.1. Ievads

2.1.1. Nolūks

Šis dokuments ir programmatūras projektējuma apraksts „GizsmoSPEED" sistēmai. Dokumenta mērķis ir detalizēti aprakstīt topošo sistēmu tā, lai tā spētu apmierināt visas prasības, kas ir izvirzītas programmatūras prasību specifikācijā, un veicinātu programmprodukta izstrādi atbilstoši PPS prasībām.

2.1.2. Darbības sfēra

Spēles un tīmekļa vietnes „GizmoSPEED" raksturojums: lidojot cauri tunelim, kas ik reizi tiek ģenerēts no jauna, spēlētājam jāizvairās no šķēršļiem un jāsakrāj pēc iespējas vairāk punktu. Pie katra spēles rezultāta, tiek piedāvāta izvēle saglabāt sakrāto punktu skaitu *MySQL* datubāzē ar iespēju aplūkot to mājas lapas rezultātu sadaļā.

Programmprodukts „GizmoSPEED" ir spēle, kas ir domāta jautrākai brīvā laika pavadīšanai. Šajā spēlē lietotāji var izmēģināt savas koncentrēšanās spējas un reakcijas ātrumu, kā arī sacensties ar draugiem, salīdzinot savus augstākos rezultātus. Produkts sastāv no divām daļām – spēles un tīmekļa vietnes.

2.1.3. Definīcijas un saīsinājumi

PPA – Programmatūras projektējuma apraksts

ER Modelis – Konceptuālais datubāzes attēlojums.

PPS – Programmatūras prasību specifikācija.

2.1.4. Saistība ar citiem dokumentiem

Dokuments veidots, balstoties uz „GizmoSPEED" programmatūras prasību specifikāciju, kā arī dokumenta izstrādes laikā ņemtas vērā vadlīnijas, kas definētas LVS 72:1996 „Ieteicamā prakse programmatūras projektējuma aprakstīšanai".

2.2. Datu struktūru un klašu projektējums

Šajā sadaļā tiek aprakstītas klases un datu struktūras, ko „GizmoSPEED" sistēma lieto spēles darbībai.

2.2.1. Datu struktūru un klašu projektējums

<table>
<tr><th colspan="3">2.2.1.1. klase UI</th></tr>
<tr><td>Klases apraksts</td><td colspan="2">Šī klase atbild par visiem lietotāja saskarnes līdzekļiem – pogām, opcijas logiem, galvenās izvēlnes animācija u.c.
Klase arī atbild par sakrāto punktu daudzumu un lietotājvārda sūtīšanu uz serveri.</td></tr>
<tr><td colspan="3">klases UI metodes</td></tr>
<tr><td>UI()</td><td colspan="2">Klases konstruktors. Definē mainīgos un izveido visu pogu objektus.</td></tr>
<tr><td>LoadContent(..)</td><td colspan="2">Ielādē visus nepieciešamos attēlus un animācijas.</td></tr>
<tr><td>Update(..)</td><td colspan="2">Šī funkcija notiek tiešraidē (tiek darbināta ik pēc katra spēles noietā kadra)
Šī funkcija atbild par pogu funkcionalitāti – veic specifiskas darbības, kad tiek nospiesta poga. Funkcija vēl atbild par galvenās izvēlnes un nāves animācijas noriti, kā arī sakrāto punktu daudzuma atjaunināšanu.</td></tr>
<tr><td rowspan="2">hashString(string)</td><td colspan="2">Šī funkcija iekodē saņemto rindu Hash kodā. Izmantoju šo funkciju, lai iekodētu sakrāto punktu daudzumu, un lietotājvārdu aizsardzības nolūkos.</td></tr>
<tr><td>Izvaddati:</td><td>string rinda Hash kodējumā</td></tr>
<tr><td rowspan="2">webPost(..)</td><td colspan="2">Šī funkcija sagatavo savienojumu un datu pārsūtīšanu uz serveri.</td></tr>
<tr><td>Izvaddati:</td><td>string servera atbilde</td></tr>
<tr><td>sendScore(..)</td><td colspan="2">Šī funkcija sūta serverim sakrāto punktu skaitu, lietotājvārdu un Hash string.</td></tr>
<tr><td>Draw(..)</td><td colspan="2">Šī funkcija nodrošina, lai visas pogas, animācijas, opcijas un iesūtīšanas logi tiek attēloti pareizajā secībā.</td></tr>
</table>

2.2.1.2. klase *Program*	
Klases apraksts	Šī klase ir īsa un atbild tikai par to, lai spēle startētu.
klases Program metodes	
Program()	Klases konstruktors. Inicializē spēlēs galveno klasi GizmoSPEEDgame.

2.2.1.3. klase *GizmoSPEEDgame*	
Klases apraksts	Šī ir galvenā spēles klase. Šajā klasē tiek definēts *ContentManager, GraphicsDevice, SpriteBatch* u.c. specifiski objekti, kuri turpmāk tiek padoti citām klasēm. Piemēram, klasei UI tiek padots *SpriteBatch* un *ContentManager,* lai šī klase spētu ielādēt animācijas attēlu, tad attēlot to uz ekrāna.
klases UI metodes	
GizmoSPEEDgame()	Klases konstruktors. Definē mainīgos un šādus objektus: • UI; • TunnelGenerator; Uzstāda *GraphicsDevice,* uzstāda noklusējuma ekrāna izšķirtspēju un novieto kursoru ekrāna centrā.
LoadContent(..)	Ielādē nepieciešamās bildes un 3D modeļus. Kā arī padod *ContentManager* citām klasēm, lai arī tās spētu ielādēt spēles sastāvdaļas.
Update(..)	Šī funkcija notiek tiešraidē. Šī funkcija ir atbildīga par: • Peles stāvokli. • Galvenā personāža atrašanās vietu, virzienu un ātrumu; • Tuneļa skaņu un to tonalitāti (*pitch,* ar katru nolidoto vienību tonalitāte paaugstinās); • Sakrāto punktu uzskati; • Spēles stāvokli (pārbauda, vai lietotājs ir zaudējis vai atrodas galvenajā izvēlnē).
HandleInput(..)	Šī funkcija atbild par tastatūras stāvokli.
UpdateCamera(..)	Šī funkcija atbild par kameru.
Draw(..)	Šī funkcija nodrošina, lai visi elementi, kas pieprasa zīmēšanu, tiek *renderēti.*

2.2.1.4. klase *Tunnel*		
Klases apraksts	Šī klase atbild par tuneļa zīmēšanu un tā saskarsmi ar galveno personāžu (*Collision Detection*).	
klases Tunnel metodes		
Tunnel(..)	Klases konstruktors. Definē mainīgos. Izveido tuneļa saskarsmes kastes (*BoundingBox*) objektu. Saskarsmes kastes ļauj uzzināt, vai personāžs ir saskāries, vai ieskrējis tuneļa sienās.	
loadTunnels(..)	Saņem visas nepieciešamos modeļus.	
drawTunnels(..)	Šī funkcija nodrošina, lai tunelis tiktu attēlots uz ekrāna.	
checkCollision(..)	Funkcija pārbauda, vai tuneļa saskarsmes kastes ir saskārušas ar personāžu.	
	Izvaddati:	*bool* true vai false;

2.2.1.5. klase *Obstacle*		
Klases apraksts	Šī klase atbild par šķēršļu zīmēšanu un to saskarsmi ar galveno personāžu (*Collision Detection*).	
klases Tunnel metodes		
Obstacle(..)	Klases konstruktors. Definē mainīgos. Izveido šķēršļa saskarsmes kastes (*BoundingBox*) objektu.	
LoadContent(..)	Saņem visas nepieciešamas modeles.	
Draw(..)	Šī funkcija nodrošina, lai šķēršļi tiktu attēloti uz ekrāna.	
checkCollision(..)	Funkcija pārbauda, vai šķēršļa saskarsmes kastes ir saskārušas ar personāžu.	
	Izvaddati:	*bool* true vai false;

2.2.1.6. klase *TunnelGenerator*		
Klases apraksts	Šī klase atbild par *Tunnel* un *Obstacle* klašu objektu veidošanu, rezultātā izveidota tuneļu un šķēršļu sistēma, caur kurām lido lietotājs. Klase vēl atbild par tuneļu un šķēršļu skaņām un palidoto tuneļu un šķēršļu dzēšanu.	
klases TunnelGenerator metodes		
TunnelGenerator()	Klases konstruktors. Definē mainīgos.	
LoadContent(..)	Ielādē visus nepieciešamos modeļus. Veido *Tunnel* un *Obstacle* objektus, rezultātā ir izveidota tuneļu un šķēršļu sistēma, caur kurām lido lietotājs.	
Update(..)	Šī funkcija notiek tiešraidē. Funkcija nosaka, kurā tunelī atrodas personāžs, dzēšot šķērsotos tuneļus un šķēršļus. Funkcija atbild par aplidoto šķēršļu skāņam un to tonalitāti (*pitch*, ar katru nolidoto vienību tonalitāte paaugstinās).	
checkColison(..)	Funkcija pārbauda, vai personāžs ir saskāries ar pašreizējām (nepalidotām) tuneļa sienām vai šķēršļiem.	
	Izvaddati:	*bool true* vai *false*
Draw(..)	Šī funkcija nodrošina, lai nepalidotie šķēršļi un tuneļi tiktu attēloti uz ekrāna.	

2.2.1.7. klase *BoundingBoxBuffers*	
Klases apraksts	Šī klase ir nepieciešama, ģenerējot saskarsmes kasti no punktu vektora. Klasei nav metodes, taču uzglabā: • *Vertices* – 3D modeļi sastāv no poligoniem, un poligonu krustpunktā ir punkts *Vertex.* • *VertexCount* • *Indices* – 3D modeļu *poligonu* veidošanā piedalās punkti *Vertex* un *Indices*. *Indices* saglāba informāciju, kā savienot punktus (*Vertex*) trīsstūrī pulksteņrādītāja virzienā, kā rezultātā ir izveidots viens *poligons*. • *PrimitiveCount*

2.2.1.8. klase *BoundingBoxComponents*		
Klases apraksts	Šī klase atbild par iesūtītā modeļa saskarsmes kastes veidošanu un tā zīmēšanu. Saskarsmes kastes zīmēšana ir paredzēta tikai izstrādātājam. Noklusētais iestatījums zīmēt kasti ir izslēgts.	
klases BoundingBoxComponents metodes		
BoundingBoxComponents()	Klases konstruktors. Definē mainīgos.	
LoadContent(..)	Saņem visus nepieciešamos 3D modeļus.	
AddVertex(..)	Pievieno Vertex punktu. Izmantota metode CreateBoundingBoxBuffers().	
CreateBoundingBoxBuffers(..)	Šī funkcija izveido saskarsmes kastes poligonus.	
	Izvaddati:	Atgriež aizpildītu iepriekš minēto 2.2.1.7. *BoundingBoxBuffers* klasi
CreateBoundingBox(..)	Šī funkcija izveido apvienoto saskarsmes kasti saņemtam modelim un nostāda to saņemtajā pozīcijā.	
	Izvaddati:	Atgriež gatavu *BoundingBox*
CreateBoundingBoxes(..)	Šī funkcija izveido neapvienotās saskarsmes kastes saņemtam modelim un nostāda tās saņemtajā pozīcijā.	
	Izvaddati:	Atgriež gatavu *BoundingBox[]*
Draw(..)	Šī funkcija nodrošina saskarsmes kastes attēlošanu uz ekrāna, ja tāds iestatījums ir ieslēgts. Noklusētais iestatījums zīmēt kasti ir izslēgts, jo šī funkcionalitāte ir nepieciešama tikai izstrādātājiem.	

2.2.1.9. klase *VertexElementExtractor*		
Klases apraksts	Šī klase sastāv no vienas metodes. Precīzāku aprakstu skat. metodēs.	
klases VertexElementExtractor metodes		
GetVertexElement(..)	Šī funkcija izvelk *Vertex* informāciju no saņemtā modeļa.	
	Izvaddati:	Vector3[] vertexData (masīvs ar *3D punktiem*)

- **Sekojošās klases ir iegādātas no interneta resursiem. Klases ir *freeware*, un tās dod iespēju atdzīvināt personāžu, pievienojot kaulu animāciju. Saite uz dotām klasēm ir pieejama interneta avotos skat. 74.lpp**

2.2.1.10. klase *SkinnedModelPipeline*	
Klases apraksts	Šī klase dod iespēju importēt modeļus ar animāciju.

2.2.1.11. klase *Keyframe*	
Klases apraksts	Apraksta katra kaula atrašanas vietu katrā kadrā.

2.2.1.12. klase *AnimationClip*	
Klases apraksts	Šī klase uzlabo visus animācijas kadrus, pārtaisot to animācijas klipā.

2.2.1.13. klase *AnimationPlayer*	
Klases apraksts	Šī klase dekodē kaulu atrašanās vietas matricu no klases *AnimationClip.*

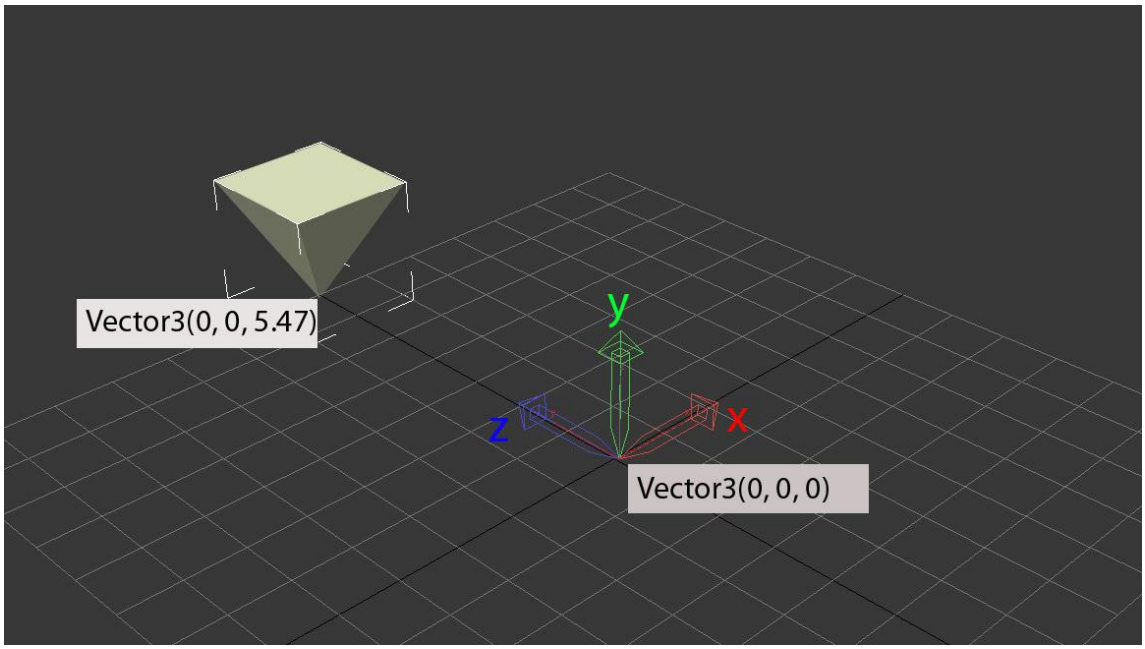
2.2.1.14. klase *SkinningData*	
Klases apraksts	Šī klase kombinē visu ievākto animācijas informāciju, lai *renderētu* (zīmētu) modeli ar kaulu animāciju.

2.3. Programmprodukta darbības koncepcija

2.3.1. Spēles sastāvdaļu zīmēšana (*renderēšana*) uz ekrāna.

Programmatūra tiek kodēta uz Visual C# ar XNA 4.0 atbalstu. Viena no lieliskajām XNA funkcijām ir iespēja viegli komunicēt ar lietotāja videokarti. Pirms renderēšanas apraksta vērts minēt vienības, kas tiek izmantotas atrašanas vietas, rotācijas u.c. uzglabāšanā. Tādu datu uzglabāšanai visērtāk ir izmantot *Vector3* - tas var saglabāt gan rotāciju, gan krāsu (*Vector3*(255,255,255) - balta krāsa), gan arī pozīciju. Vector3(x,y,z) ir punkts, kas atrodas spēles visumā.

2.3.1.1. Vector3 attēls



Pieņemsim, ka ir jāizveido galvenā varoņa pozīcija. To var panākt ar sekojošo kodu:

```
Vector3 playerPositon1 = new Vector3(0,0,0);
```

Kodējot spēli, izmantot vektorus ir ļoti vienkārši un parocīgi. Spēles programmēšanas gaitā pārsvara tiek izmantoti vektori, taču šis vektors vēl nav gatavs iesūtīšanai uz videokarti. Videokarte *nesaprot* vektorus, tā vietā sagaida matricu. Tādēļ ļoti labi, ka XNA piedāvā ērtu funkciju (Matrix.CreateTranslation(Vector3)), kas konvertē vektoru uz matricu:

```
Vector3 playerPositon1 = new Vector3(0,0,0);
Matrix playerPositonMatrix = Matrix.CreateTranslation(playerPosition1);
```

spēles dzinis nenoskaidros, kāda ir pieprasītā izšķirtspēja, kāds ir pieprasītais projicējums un kur attiecība pret ekrānu ir jāzīmē 3D attēls. Kamera ir nozīmīga katra objekta attēlošanā uz ekrānā, jo katram 3D objektam ir jāpielāgojas pie šādiem nosacījumiem:

- Objekta attēlošanas skatpunkts. Skatpunkts - kameras atrašanas vieta un orientācija.
- Kameras attēla projekcija un ekrāna izšķirtspēja.
  (skat. attēlu 2.3.1.2)

Katram 3D objektam ir jāpadod kameras dati, lai tas spētu veiksmīgi attēloties uz ekrāna. Sekojošais kods izveido kameras skatu un projekciju:

Skatu viedo no kameras atrašanās vietas, rotācijas un orientācijas.

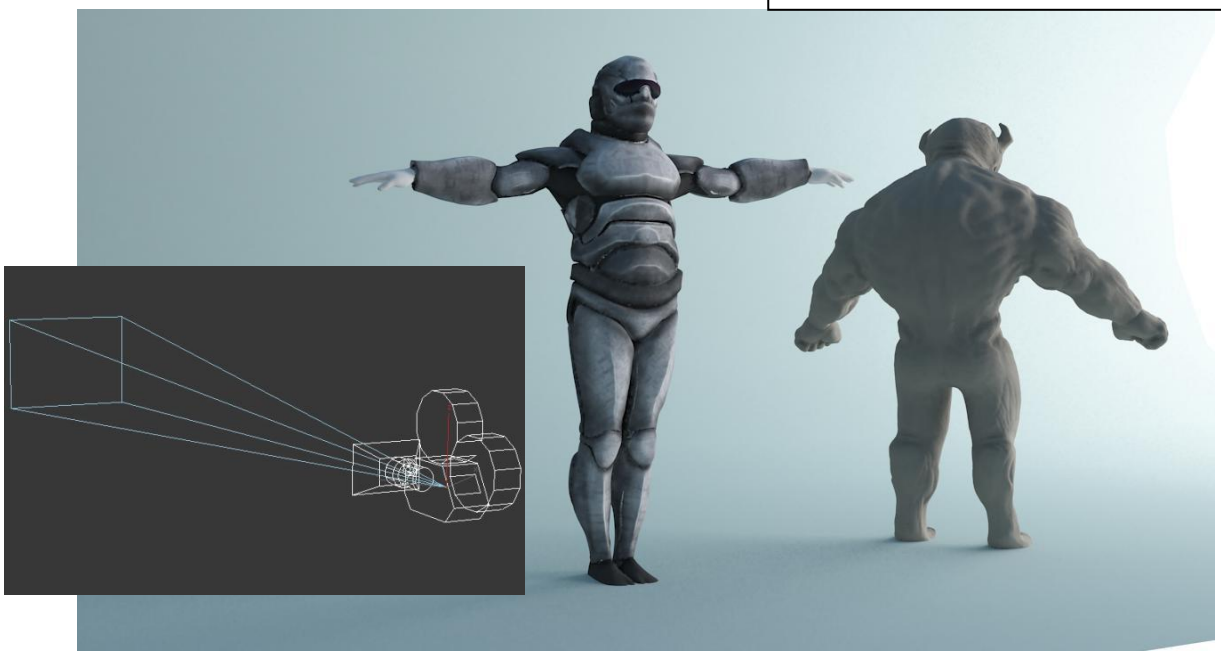
```
Matrix view = Matrix.CreateTranslation(cameraPosition) *
                              Matrix.CreateRotationY(MathHelper.ToRadians(cameraRotation)) *
                              Matrix.CreateRotationX(MathHelper.ToRadians(cameraArc)) *
                              Matrix.CreateLookAt(new Vector3(0, 0, -cameraDistance),
                                                  new Vector3(0, 0, 0), Vector3.Up);

Matrix projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.ToRadians(75.0f),
                                             device.Viewport.AspectRatio, 1, 10000);
```

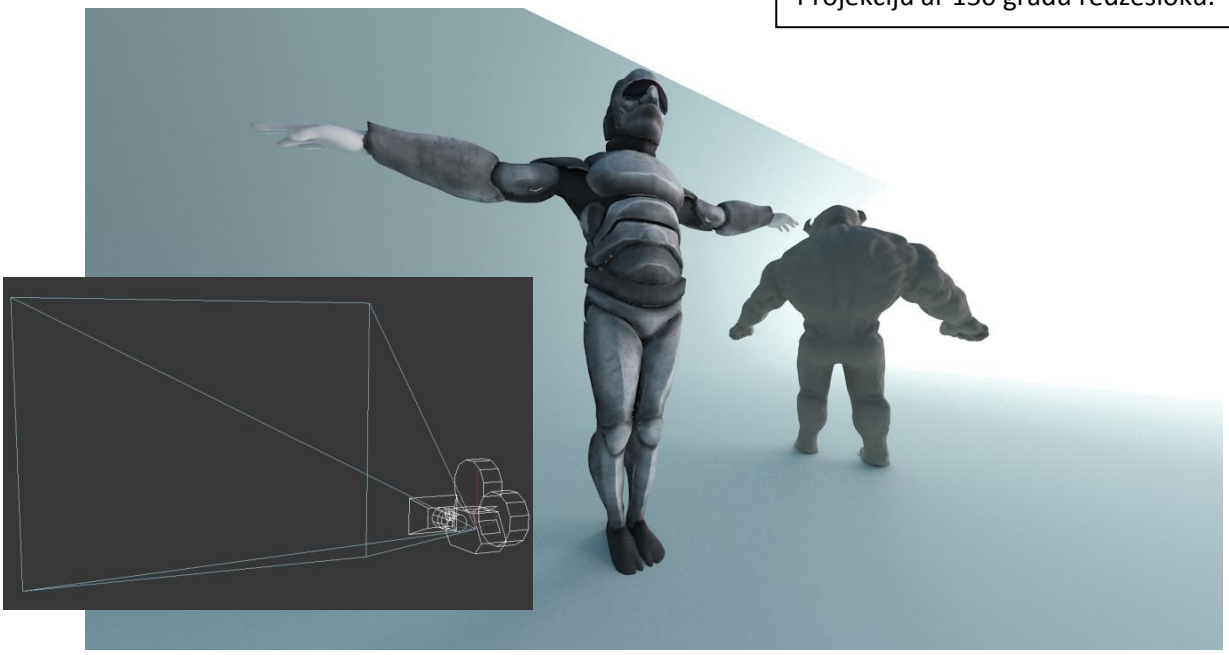
Projekciju veido no redzesloka (75 grādi) un ekrāna izšķirtspējas proporcijas.

2.3.1.2 Dažādu projekciju attēli

Projekciju ar 45 grādu redzesloku.



Projekciju ar 150 grādu redzesloku.



Rezultātā ir pieejama varoņa atrašanās vietas matrica, kameras skata un projekcijas matrica. Atliek tikai padot nepieciešamo informāciju videokartei un izsaukt renderēšanas metodi.

```
foreach (ModelMesh mesh in playerModel.Meshes)
                {
                    foreach (BasicEffect effect in mesh.Effects)
                    {
                        effect.EnableDefaultLighting();

                        effect.World = playerPositionMatrix;
                        effect.View = view;
                        effect.Projection = projection;
                    }
                    mesh.Draw();
                }
```

Varoņa atrašanās vietas matrica.

Kameras skatpunkta matrica.

Kameras projekcijas matrica.

Attēlot varoni.

2.3.1.3 Spēļu sastāvdaļu renderēšanas attēls

2.3.2. Varoņa saskarsme ar tuneļa sienām vai šķēršļiem.

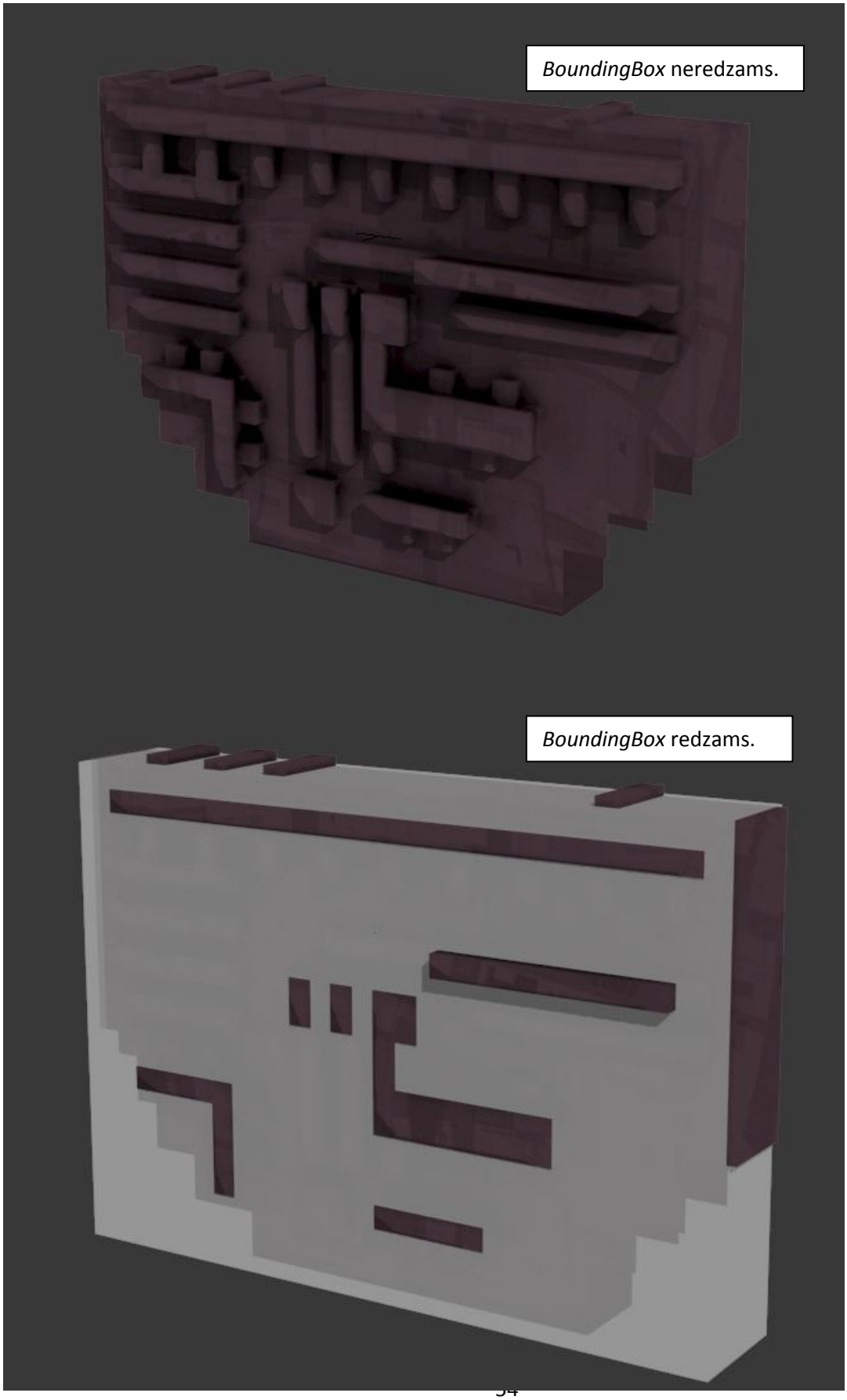
Lai iegūtu lielu punktu skaitu, spēlētājam ir jāizvairās no šķēršļiem un tuneļa sienām. Bez saskarsmes pārbaudes spēle zaudē savu būtību. Tādēļ atradu efektīvu risinājumu, kuru īsumā aprakstīšu šajā nodaļā.

2.3.2.1 Saskarsmes pārbaude.

Sākotnējā doma bija sastādīt saskarsmes pārbaudes algoritmu, kas pārbauda, vai punkts ir pieskāries kādam no modeļa *poligoniem*. Algoritmam ir savi **plusi**, piemēram, sadursmes pārbaude ar animētiem šķēršļiem. Taču šāda algoritma kodēšana ir sarežģīta un pilna ar negaidītiem *pārsteigumiem*, kuru risināšana ir laikietilpīgs process.

Tuneļi un šķēršļi ir diezgan viendabīgi. Spēle sastāv no diviem tuneļiem un deviņiem šķēršļiem. Sapratu, ka efektīvāka tehnika risināt sadursmes pārbaudi ir *BoundingBox* jeb *HitBox* metode. Precīzāk, sākumā katram modelim tiek izveidots neredzams paralelograms, kas ir pilnība apvilkts ap dotā modeļa apkārtmēra robežām. Pēc pieprasījuma var pārbaudīt sadursmi - neredzamais paralelograms pārbauda, vai tajā atrodas dotais punkts. Ja salīdzina šo sadursmes pārbaudi ar iepriekšējo, tad šī pieeja patērē daudz mazāk datora resursus. *BoundingBox* piemēru var aplūkot skat. 2.3.2.2. attēlā.

2.3.2.2 *BoundingBox* vizualizācija.



2.3.3. Rezultāta iesūtīšana datu bazē ar aizsardzību.

No spēles klienta Lietotājvārda un sakrāto punktu daudzuma sūtīšana uz serveri notiek ar *POST* metodi. Programmatūras prasību specifikācija pieprasa, lai sūtīšana būtu aizsargāta. Izlēmu lietot *Hash* kodējumu, lai nodrošinātu sūtīto datu drošību. Princips ir šāds – tiek izveidota parasta rinda (*string*), kurā iekļauts lietotājvārds un sakrāto punktu daudzumu, tad izveidota vēl viena rinda, kas satur lietotājvārdu, punktu daudzumu un slepeno paroli. To iesūtu uz 2.2.1. nodaļā aprakstīto metodi *hashString(),* kas pārvērš rindu *Hash* kodējumā. Veicot minētās darbības, apvienoju abas rindas. Rezultātā rinda, ko izsūtu ar *POST* metodi, satur:

- Lietotājvārdu;
- Sakrāto punktu daudzumu;
- *Hash* kodu.

Uz servera atrodas PHP skripts, kas, izmantojot saņemto lietotājvārdu, sakrāto punktu skaitu un servera slepeno paroli (identiska ar klienta slepeno paroli), ģenerē savu *Hash string,* kas tiek salīdzināta ar iesūtīto. Ja slepenās paroles ir identiskas, tad serveris veic pieprasījumu ierakstīt jauno rezultātu datubāzē.

```
public void sendScore( string name,  int score)
        {
            string highscoreString = name + score +
            string postString = "&Name=" + name + "&Score=" + score + "&Hash=" +
h                                          +  hashString(highscoreString);
            string response = null;
            response = webPost("http://25.153.234.96/score.php", postString);
        }
```

Rezultāta iesūtīšana uz serveri.

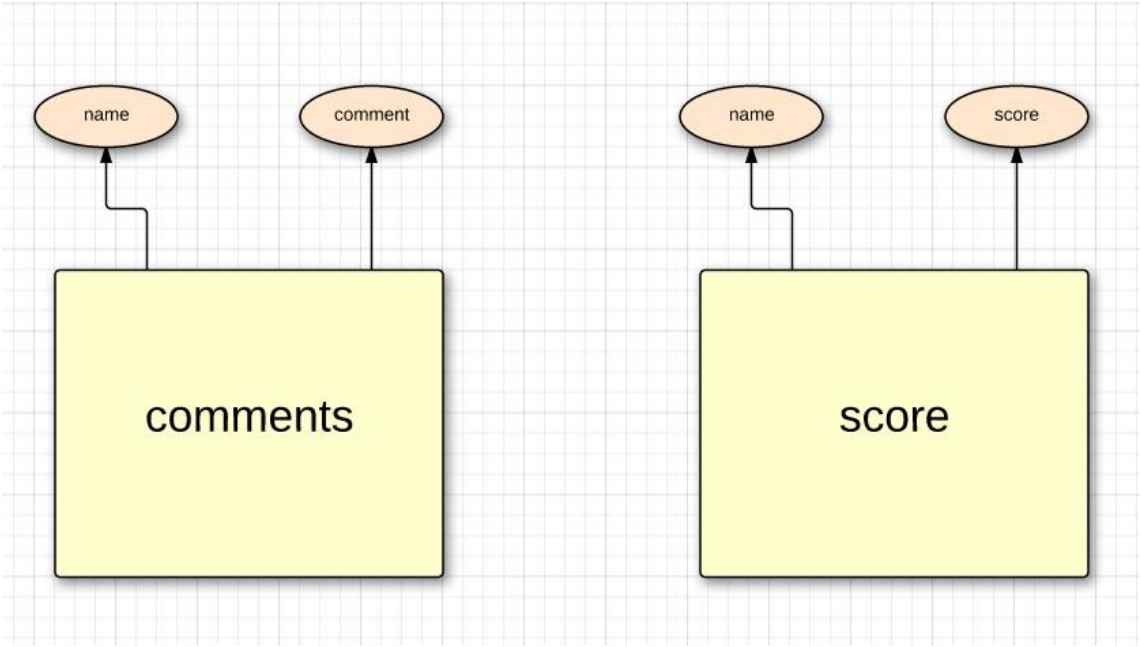
```
private string hashString(string _value)
        {
            System.Security.Cryptography.MD5CryptoSe
System.Security.Cryptography.MD5CryptoServiceProvide

            byte[] data = System.Text.Encoding.ASCII.GetBytes(_value);
            data = x.ComputeHash(data);
            string ret = "";
            for (int i = 0; i < data.Length; i++) ret +=
data[i].ToString("x2").ToLower();
            return ret;
        }
```

Rindas pārvēršana *Hash* kodējumā.

2.4. Datubāzes projektējums

2.4.1. Konceptuālais ER modelis



2.4.2. Datubāzes tabulas

2.4.2.1 tabula „score”			
Nosaukums	Datu tips	Apraksts	Piezīmes
id	int(10)	Lietotāja identifikators	PK, AI
name	varchar(100)	Lietotājvārds	utf8_unicode_ci
score	int(11)	Sakrāto punktu daudzums	
created_at	timestamp	Pievienotā ieraksta datums un laiks	
updated_at	timestamp	Atjauninātā ieraksta datums un laiks	

2.4.2.2 tabula „comments”			
Nosaukums	Datu tips	Apraksts	Piezīmes
id	int(10)	Lietotāja identifikators	PK, AI
name	varchar(100)	Lietotājvārds	utf8_unicode_ci
comment	varchar(5000)	Lietotāja atsauksme	utf8_general_ci
created_at	timestamp	Pievienotā ieraksta datums un laiks	
updated_at	timestamp	Atjauninātā ieraksta datums un laiks	

3. Testēšanas dokumentācija

3.1. Ievads

Testēšanas dokumentācijā ir aprakstītas testēšanas metodes un rezultāti.

3.2. Nolūks

Šī dokumenta nolūks ir aprakstīt testēšanas norisi un sasniegtos rezultātus. Dokuments domāts programmprodukta pasūtītājiem, izstrādātā darba pārbaudei un neatkarīgiem testētajiem.

3.3. Testu sagatavošana

Izstrādes laikā testēšana bija veikta pēc *baltās kastes* metodes, kad jau ir zināma informācija par programmas uzbūvi un algoritmiem. Tā tika veikta pēc katra jauna moduļa izstrādes ar nolūku izlabot jauniegūtās kļūdas.

Izstrādes beigās, testēšanas procesam pieslēdzās arī testētāji, kas veica testēšanu pēc *melnās kastes* metodes, nezinot spēles loģiku vai struktūru, tie orientējās tikai pēc vizuālā produkta prasībām un izvirzītiem mērķiem.

3.4. Testēšanas plāns

Šajā nodaļa ir aprakstīti visi testi un to rezultāti. Testi variē no svarīgākajiem spēles aspektiem līdz mazsvarīgajām funkcijām, kā, piemēram, spēles atkārtota izspēlēšana 15 reizes pēc kārtas.

3.5 Tabula Testēšanas žurnāls

Testa apraksts	Pareiza izpilde	Testa rezultāts
1. Lietotājs ieslēdz spēli	Spēle tiek ieslēgta, uz ekrāna parādoties izvēlnei.	✔
1. Spēlētājs ieslēdz spēli 2. Atver iestatījumu sadaļu 3. Maina ekrāna izšķirtspēju uz 1980x1080	• Spēles klients pielāgojas izvēlētai izšķirtspējai • Spēle atgriež kļūdas paziņojumu, ja ekrāns neatbalsta tādu izšķirtspēju.	✔

1. Spēlētājs ieslēdz spēli 2. Spēlētājs spiež pogu „Exit”.	Spēles klients tiek izslēgts.	✔
1. Spēlētājs ieslēdz spēli 2. Spēlētājs sāk spēlēt spēli 3. Spēlētājs ietriecas tuneļa sienā	Spēle tiek pārtraukta. Tiek attēlota nāves animācija un skaņa, tad tiek attēlots sakrāto punktu daudzums, kā arī „Menu”, „Replay” un „Submit” pogas.	✔
1. Spēlētājs ieslēdz spēli 2. Spēlētājs sāk spēlēt spēli līdz brīdim, kad ietriecas kādā no tuneļa sienām vai šķēršļiem. 3. Izvēlās sākt spēli no jauna.	Spēle tiek atsākta, bet ar citādi ģenerētiem tuneļiem un šķēršļiem.	✔
1. Spēlētājs ieslēdz spēli 2. Spēlētājs sāk spēlēt spēli 3. Ietriecas tuneļa sienā ar 90° leņķi (perpendikulāri tuneļa sienai)	Spēle tiek pārtraukta. Tiek attēlota nāves animācija un skaņa, tad tiek attēlots sakrāto punktu daudzums, kā arī „Menu”, „Replay” un „Submit” pogas.	✔
1. Spēlētājs ieslēdz spēli 2. Spēlētājs sāk spēlēt spēli līdz brīdim, kad ietriecas kādā no tuneļa sienām vai šķēršļiem. 3. Izvēlās sākt iziet uz galveno izvēlni 4. Maina izšķirtspēju uz 1280x720	Spēles klients pielāgojas izvēlētai izšķirtspējai.	✔
1. Spēlētājs ieslēdz spēli 2. Ieskrien šķērslī un izvēlas sākt spēli no	Katru reizi tiek atsākta spēle, bet ar citādi ģenerētiem tuneļiem un šķēršļiem.	✔

jauna. Atkārto to 15 reizes		
1. Spēlētājs ieslēdz spēli 2. Ietriecas slepenā tuneļa sienā.	Spēle tiek pārtraukta. Tiek attēlota nāves animācija un skaņa, tad tiek attēlots sakrāto punktu daudzums, kā arī „Menu”, „Replay” un „Submit” pogas.	✔
1. Spēlētājs ieslēdz spēli 2. Spēlē līdz ietriecas kādā no šķēršļiem, sakrājot pēc iespējas vairāk punktu 3. Izvēlas iesūtīt sakrāto punktu daudzumu uz serveri. 4. Ieraksta savu lietotājvārdu un spiež „Submit”	Datu bāze ir pievienots jauns ieraksts, kas satur sakrāto punktu daudzumu un lietotājvārdu. Šis rezultāts ir arī atspoguļots tīmekļa vietnes rezultātu sadaļā.	✔
1. Spēlētājs ieslēdz spēli 2. Spēlētājs vairākas reizes izspēlē spēli un iesūta sakrāto punktu daudzumu un lietotājvārdu uz serveri, neizejot no spēles.	Visi iesūtītie rezultāti tiek pievienoti datu bāzei. Šie rezultāti atspoguļoti tīmekļa vietnes rezultātu sadaļā.	✔

4. Kvalitātes nodrošināšana

Lai nodrošinātu kvalitāti dokumentu izstrādē, PPS un PPA tika izstrādāti, balstoties uz Latvijas Valsts standartiem – LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis" un LVS 71:1996 „Ieteicamā prakse programmas projektējuma aprakstīšanai". Kods ir loģiski strukturēti sadalīts klasēs un funkcijās, lai būtu to ērti lasīt un uztvert un, protams, atbilstoši komentēts. Programmatūras izstrāde tika veikta objektorientētā programmēšanas stilā, kas atviegloja darbu jaunu moduļu ieviešanai. Funkcijas tika veikti testi, kā arī tika testēta sistēmas kopēja darbība un veikta rūpīga atkļūdošana. Spēles darbība tika pārbaudīta uz vairākām darba stacijām, rezultātā tika veiktas programmprodukta optimizācijas, lai labāk darbotos spēle.

5. Konfigurācijas pārvaldība

Tā kā programmas kods netika izstrādāts cita projekta ietvaros, šī projekta konfigurācijas pārvaldībai netika lietota versiju pārvaldības sistēma. Tā vietā spēles izstrādes procesā tika saglabātas dažādas projekta versiju rezerves kopijas uz man pieejamā *ftp* servera un cietiem diskiem.

6. Darbietilpības novērtējums

Darbietilpības novērtējums tika veikts, izmantojot COCOMO 81 metodi. Šī metode ļauj aprēķināt aptuveno darbietilpību personmēnešos, izmantojot formulu E=a*KloCb*DRF, kur KloC apzīmē prognozējamo koda rindu skaitu tūkstošos, DRF apzīmē darbietilpību regulējošo faktoru, kas tiek aprēķināts no vairāku faktoru novērtējuma un b ir koeficenti, kas atkarīgi no projekta izstrādes tipa.

Visus darbietilpību regulējošos faktorus samērā adekvāti spēju novērtēt pats. Lai uzzinātu prognozējamo koda rindiņu skaitu, saskaitīju visas paša rakstīto kodu, kā arī paša rakstīto kodu tīmekļa vietnei. Tīmekļa vietne aizņem 691 koda rindas, savukārt spēle aizņem 2766 koda rindas. Kopējais rindu skaits ir 3457, līdz ar to KloC = 3.457.

6.1. Darbietilpību regulējošā faktora noteikšana:

- Produkta atribūti
    - Pieprasīta izturība – ļoti maza(0.75)
    - Datu bāzes izmērs – ļoti mazs(0.94)
    - Sistēmas sarežģītība – vidējas (0.70)
- Tehnikas atribūti
    - Izpildes laika ierobežojumi – vidēji(1.00)
    - Atmiņas ierobežojumi – mazi(1.00)
    - Vides mainība – maza(0.87)
    - Nepieciešamais laiks programmas izpildei – mazs(0.87)
- Personālie atribūti
    - Analītiskās spējas – vidējas(1.00)
    - Programmatūras izstrādes pieredze – ļoti maza (1.29)
    - Programmētāja izstrādes spējas – liela(0.86)
    - Izstrādes vides pieredze – liela(0.90)
    - Programmēšanas valodas pieredze – vidēja(1.00)
- Projekta atribūti
    - Moderno programmēšanas paņēmienu izmantošana– vidējas(1.00)
    - Izstrādes rīku lietojums – liels(1.11)
    - Izstrādes laika ierobežojumi – vidēji(1.0)

Sareizinot visus iepriekš iegūtos koeficentus, varam iegūt Darbietilpības regulējošo faktoru, jeb DRF, kas ir vienāds ar 0.4139. Projekta tips pēc COCOMO iedalījuma ir „organisks", no kā izriet sekojoši koeficenti, a = 3.2 un b=1.05.

Izejot no iepriekšējā vērtējuma varam secināt, ka darbietilpība, jeb $E=a*Kloc^{b}*DRF=3.2*3.457^{1.05}*0.4139=4,607$ personmēneši.

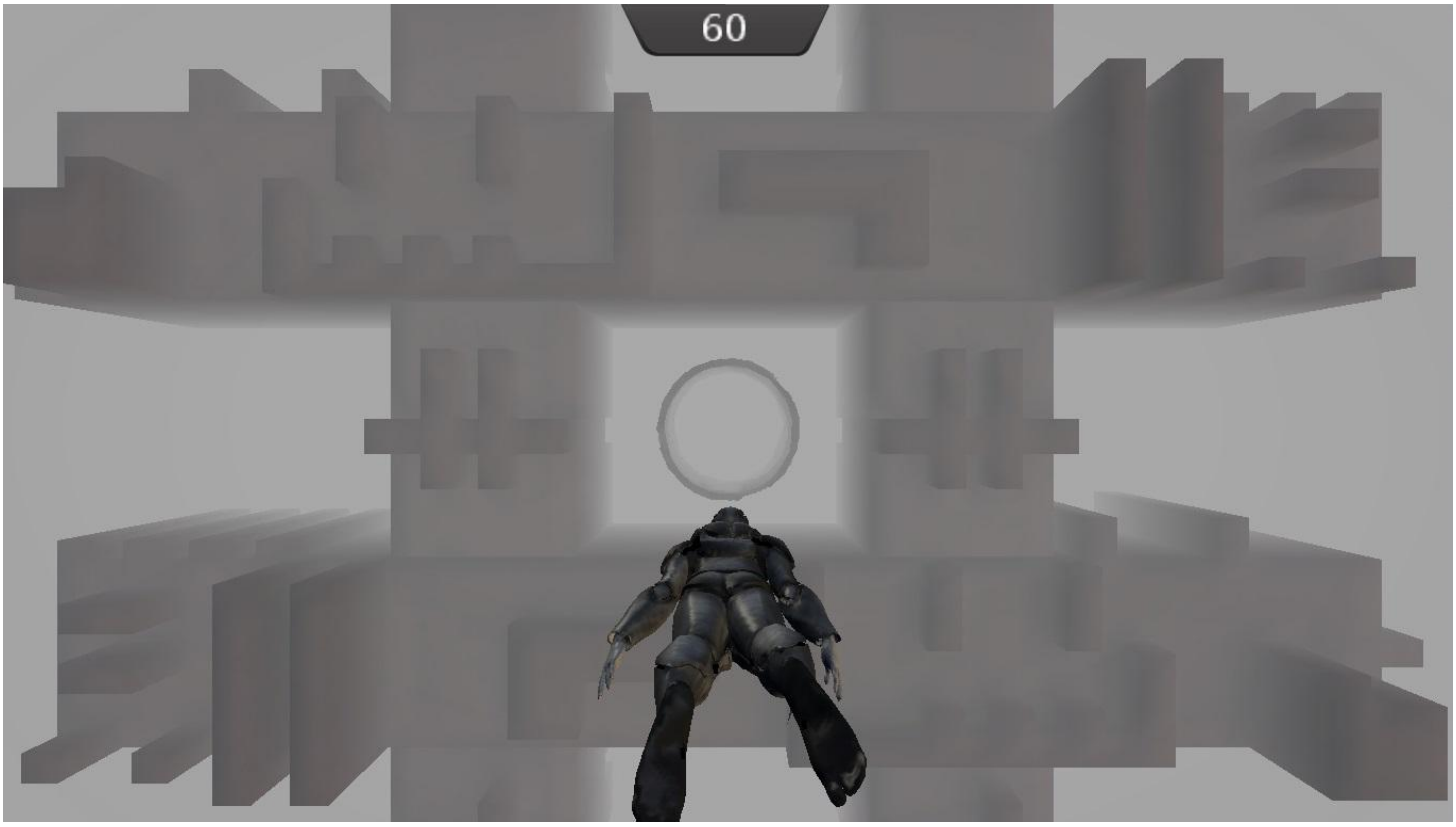
Pēc programmatūras izstrādes var secināt, ka darbietilpības faktoru izvēle bija pietiekami precīza un šī metode ir pietiekami labi piemērota šāda projekta darbietilpības novērtēšanai.

7. Ekrānuzņēmumi

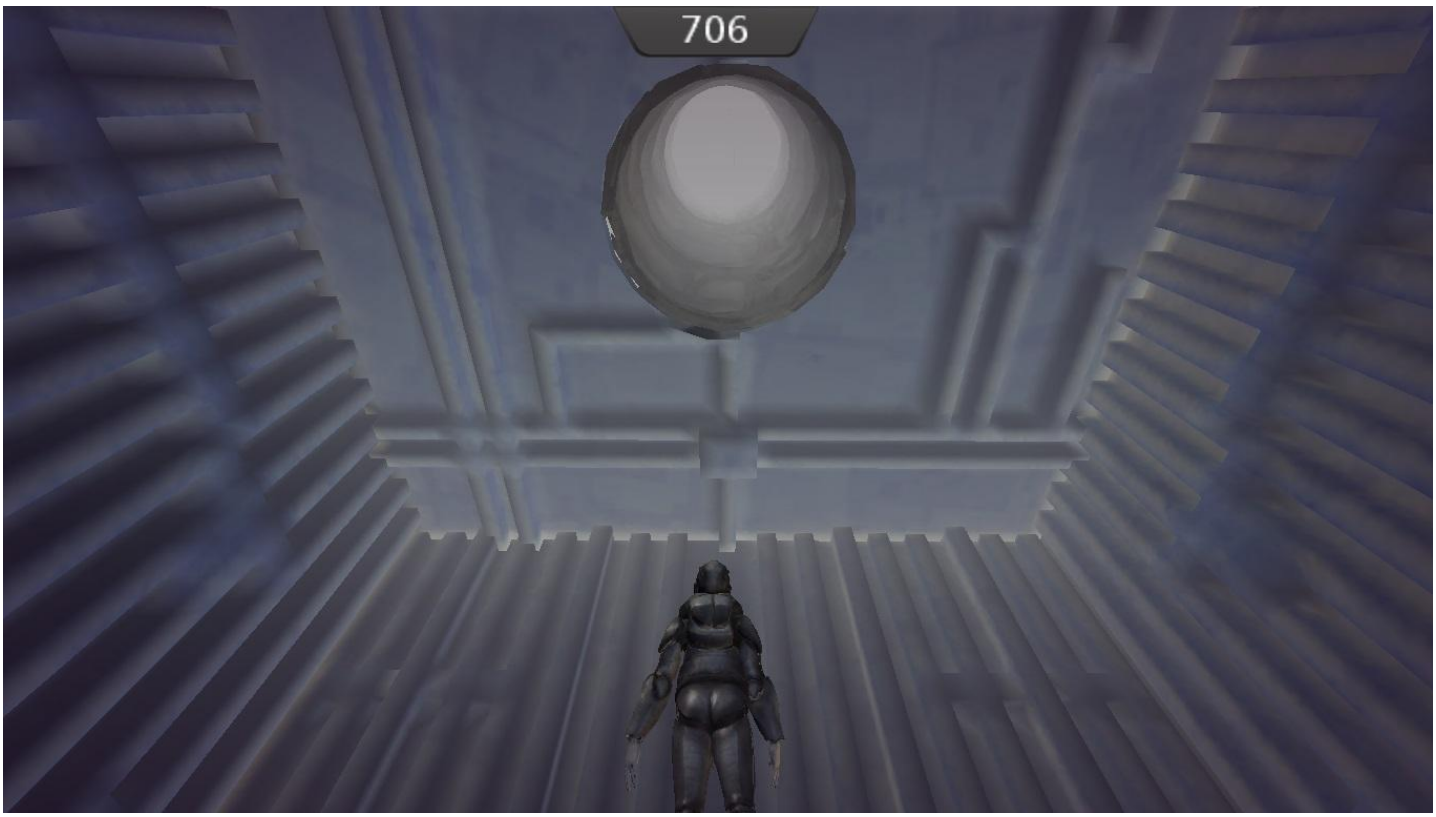
8.1 ***Attēls Spēle „GizmoSPEED”***

7.1 *Attēls Galvenā izvēlne*

7.2 Attēls spēles sākums



7.3 Attēls slepenais tunelis

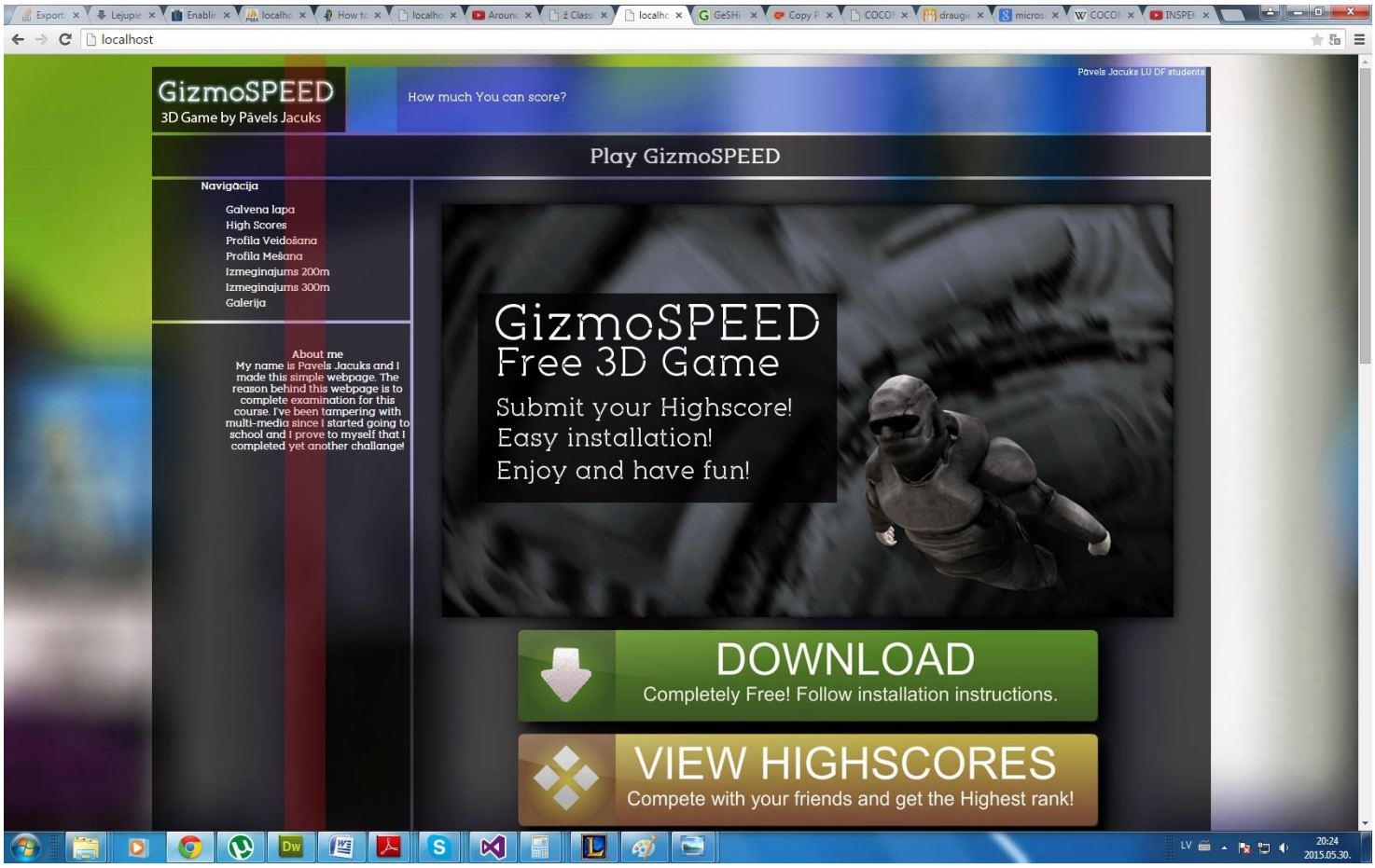


7.4 Attēls varoņa izvairīšanās no šķēršļiem

7.5 Attēls opcijas pēc zaudētās spēles

7.6 Attēls galvenais varonis

7.7 Attēls „GizmoSPEED” tīmekļa vietne



8. Secinājumi

Darba laikā guvu neaizvietojamu pieredzi reālas un lietojamas spēles izstrādē. Pilnvērtīgāk apguvu C# programmēšanas valodu Visual Studio izstrādes vidē. Ieguvu vispārēju priekšstatu par dažādu algoritmu optimizāciju. Guvu padziļinātāku priekšstatu par objektorientēto programmēšanas pieeju. Rakstot darbu, esmu guvis nozīmīgu pieredzi kvalitatīvas dokumentācijas izstrādāšanā. Modelējot, apstrādājot spēlei vajadzīgās sastāvdaļas - modeļi un animācijas –, iemācījos pavisam jaunas metodes, kā sasniegt vēlamo rezultātu. Autors ir sasniedzis uzstādītos mērķus un izpildījis izvirzītos uzdevumus, un uzskata, ka spēle „GizmoSPEED" un izveidotā tīmekļa vietne ir izdevušies veiksmīgi.

9. Izmantotā literatūra un avoti

1. LVS 68:1996 „Programmatras prasību specifikācijas ceļvedis”
2. LVS 72:1996 „Ieteicama prakse programmatūras projektējuma aprakstīšanai”
3. Animating in XNA 4.0 - http://xbox.create.msdn.com/education/catalog/tutorial/skinned_model_extensions
4. Reimer’s XNA Tutorials - http://www.riemers.net/eng/Tutorials/XNA/Csharp/series2.php

10. Pielikums

```
namespace GizmoSPEED
{
    class UI //THE UI. 2D gaphics like buttons and graphic animations.
    {
        SpriteBatch spriteBatch;
        Texture2D vignette;
        Texture2D scoreBoard;
        Texture2D enterName;
        SpriteFont scoreFont;
        KeyboardState keyBoardState;
        KeyboardState oldKeyBoardState;
        bool enterNameOn = false;

        VideoPlayer videoPlayer;
        Video menuVideo;

        Texture2D optionWindow;
        Texture2D pleaseWait;
        bool showOptions = false;
        public bool gameOver = false;
        public bool gameStarted = false;



        Button play;
        Button options;
        public Button exit;
        Button exitOptions;
        public Button applyOptions;
        Button invisible;
        Button invisible2;

        public Button deathQuit;
        public Button deathReplay;
        Button deathSubmit;
        Texture2D deathOptions;


        public Button enterSubmit;
        Button enterBack;

        public int currRes = 6;

        int i = 0;
        int g = 0;
        double time = 0;
        double time2 = 0;

        Texture2D[] menu = new Texture2D[39];
        Texture2D[] deathScreen = new Texture2D[55];




        string score;
        public bool mouseReleased = true;


        float vignetteAlpha;

        int screenResolutionX;
        int screenResolutionY;
```

```
        public int[,] resolution;
        string resolutionString;
        string animationString = "ON";
        public bool animation = true;

        bool showPleaseWait = false;
        SoundEffect soundDeathEerie;
        SoundEffectInstance soundDeathEerieInstance;

        bool deathPlayedOnce = true;
        string scoreName = "Name";
        public UI(int screenX, int screenY)
        {
            vignetteAlpha = 0.5f;
            ///1.2f
            ///


            resolution = new int[9, 2];

            resolution[0, 0] = 800;
            resolution[0, 1] = 600;
            resolution[1, 0] = 1152;
            resolution[1, 1] = 864;
            resolution[2, 0] = 1400;
            resolution[2, 1] = 1050;
            resolution[3, 0] = 1600;
            resolution[3, 1] = 1200;
            resolution[4, 0] = 1024;
            resolution[4, 1] = 600;
            resolution[5, 0] = 1280;
            resolution[5, 1] = 720;
            resolution[6, 0] = 1366;
            resolution[6, 1] = 768;
            resolution[7, 0] = 1680;
            resolution[7, 1] = 1050;
            resolution[8, 0] = 1920;
            resolution[8, 1] = 1080;

            resolutionString = resolution[6,0].ToString() + "x" + resolution[6,1].ToString();

            screenResolutionX = screenX;
            screenResolutionY = screenY;
            videoPlayer = new VideoPlayer();
           // videoPlayer.Play(menuVideo);
            play = new Button("PLAY", new Vector2(30, 420), false);
            options = new Button("OPTIONS", new Vector2(30, 520), false);
            exit = new Button("EXIT", new Vector2(30, 620), false);

            exitOptions = new Button("BACK", new Vector2(screenResolutionX / 2, screenResolutionY
/ 2 - 20), true);
            applyOptions = new Button("APPLY", new Vector2(screenResolutionX / 2 + 170,
screenResolutionY / 2 - 20), true);
            invisible = new Button("47 F*CKS", new Vector2(screenResolutionX / 2 + 150,
screenResolutionY / 2 - 147), true);
            invisible2 = new Button("47 F*CKS", new Vector2(screenResolutionX / 2 + 150,
screenResolutionY / 2 - 80), true);


            deathQuit = new Button("MENU", new Vector2(screenResolutionX / 2 - 290,
screenResolutionY / 2 + 30), true);
            deathReplay = new Button("REPLAY", new Vector2(screenResolutionX / 2 - 85,
screenResolutionY / 2 + 30), true);
```

```
            deathSubmit = new Button("SUBMIT", new Vector2(screenResolutionX / 2 + 120,
screenResolutionY / 2 + 30), true);

            enterBack = new Button("BACK", new Vector2(screenResolutionX / 2 - 195,
screenResolutionY / 2 - 47), true);
            enterSubmit = new Button("SUBMIT", new Vector2(screenResolutionX / 2 + 75,
screenResolutionY / 2 - 47), true);


            score = "47";
        }
        public void LoadContent(ContentManager Content, SpriteBatch spBatch)        {
            spriteBatch = spBatch;
            vignette = Content.Load<Texture2D>("vignette");//load from content folder
            scoreBoard = Content.Load<Texture2D>("scoreBoard");
            scoreFont = Content.Load<SpriteFont>("font");
            optionWindow = Content.Load<Texture2D>("Options");
            deathOptions = Content.Load<Texture2D>("scoreSubmit");
            pleaseWait = Content.Load<Texture2D>("pleaseWait");
            enterName = Content.Load<Texture2D>("EnterName");

            play.LoadContent(Content, spriteBatch);
            options.LoadContent(Content, spriteBatch);
            exit.LoadContent(Content, spriteBatch);

            exitOptions.LoadContent(Content, spriteBatch);
            applyOptions.LoadContent(Content, spriteBatch);
            invisible.LoadContent(Content, spriteBatch);
            invisible2.LoadContent(Content, spriteBatch);


            deathQuit.LoadContent(Content, spriteBatch);
            deathReplay.LoadContent(Content, spriteBatch);
            deathSubmit.LoadContent(Content, spriteBatch);

            enterBack.LoadContent(Content, spriteBatch);
            enterSubmit.LoadContent(Content, spriteBatch);

            soundDeathEerie = Content.Load<SoundEffect>("Audio\\Gleams_05");
            soundDeathEerieInstance = soundDeathEerie.CreateInstance();
            soundDeathEerieInstance.IsLooped = false;
            soundDeathEerieInstance.Volume = 0.09f;

            for (int j = 0; j <= 38; j++)
                menu[j] = Content.Load<Texture2D>("Menu\\"+j.ToString());

            for (int j = 0; j <= 54; j++)
                deathScreen[j] = Content.Load<Texture2D>("Death\\" + j.ToString());

        }

        private string webPost(string _URI, string _postString) //create connection with server

        {
            const string REQUEST_METHOD_POST = "POST";
            const string CONTENT_TYPE = "application/x-www-form-urlencoded";
            Stream dataStream = null;
            StreamReader reader = null;
            WebResponse response = null;
            string responseString = null;

            WebRequest request = WebRequest.Create(_URI);

            request.Method = REQUEST_METHOD_POST;
```

```
            string postData = _postString;
            byte[] byteArray = Encoding.UTF8.GetBytes(postData);

            request.ContentType = CONTENT_TYPE;

            request.ContentLength = byteArray.Length;

            dataStream = request.GetRequestStream();

            dataStream.Write(byteArray, 0, byteArray.Length);

            dataStream.Close();
            return responseString;
        }

        private string hashString(string _value)
        {
            System.Security.Cryptography.MD5CryptoServiceProvider x = new
System.Security.Cryptography.MD5CryptoServiceProvider();
            byte[] data = System.Text.Encoding.ASCII.GetBytes(_value);
            data = x.ComputeHash(data);
            string ret = "";
            for (int i = 0; i < data.Length; i++) ret += data[i].ToString("x2").ToLower();
            return ret;
        }

        public void sendScore( string name,  int score)
        {
            string highscoreString = name + score + "Aizsardziba";
            string postString = "&Name=" + name + "&Score=" + score + "&Hash=" +
hashString(highscoreString);
            string response = null;
            response = webPost("http://25.153.234.96/score.php", postString);
            //return response.Trim();
        }

        public void Update(int scre, GameTime gameTime, MouseState m, bool gameOvr, KeyboardState
kstate) //this is realtime function
        {
            oldKeyBoardState = keyBoardState;
            keyBoardState = kstate;
            gameOver = gameOvr;
            score = scre.ToString();
            vignetteAlpha = float.Parse(score) / 1.5f / 1000;
            if (m.LeftButton == ButtonState.Released) mouseReleased = true;
            else mouseReleased = false;
            play.Update(m);
            options.Update(m);
            exit.Update(m);

            if (!gameStarted && play.clicked)
            {
                gameStarted = true;
                play.clicked = false;
            }
            if (options.clicked)
            {
                //Debug.Print("tre");
                showOptions = true;
                options.clicked = false;
            }
            if (showOptions) //toggles wether options should be showed
            {
```

```
                exitOptions.Update(m);
                applyOptions.Update(m);
                invisible.Update(m);
                invisible2.Update(m);

                if (invisible.clicked && mouseReleased) //changing the resolution
                {
                    mouseReleased = false;
                    if (currRes == 8)
                        currRes = 0;
                    else
                        currRes++;

                    resolutionString = resolution[currRes, 0].ToString() + "x" +
                    resolution[currRes, 1].ToString();
                }
                if (invisible2.clicked && mouseReleased) //changin animation in options
                {
                    mouseReleased = false;
                    if (animationString == "ON")
                    {
                        animationString = "OFF";
                        animation = false;
                    }
                    else
                    {
                        animationString = "ON";
                        animation = true;
                    }
                }

                if (applyOptions.clicked)
                {
                    screenResolutionX = resolution[currRes, 0];
                    screenResolutionY = resolution[currRes, 1];
                    play.UpdatePosition(new Vector2(30, screenResolutionY / 2 + 45));
                    options.UpdatePosition(new Vector2(30, screenResolutionY / 2 + 145));
                    exit.UpdatePosition(new Vector2(30, screenResolutionY / 2 + 245));
                    exitOptions.UpdatePosition(new Vector2(screenResolutionX / 2,
screenResolutionY / 2 - 20));
                    applyOptions.UpdatePosition(new Vector2(screenResolutionX / 2 + 170,
screenResolutionY / 2 - 20));
                    invisible.UpdatePosition(new Vector2(screenResolutionX / 2 + 150,
screenResolutionY / 2 - 147));
                    invisible2.UpdatePosition(new Vector2(screenResolutionX / 2 + 150,
screenResolutionY / 2 - 80));

                    deathQuit.UpdatePosition(new Vector2(screenResolutionX / 2 - 290,
screenResolutionY / 2 + 30));
                    deathReplay.UpdatePosition(new Vector2(screenResolutionX / 2 - 85,
screenResolutionY / 2 + 30));
                    deathSubmit.UpdatePosition(new Vector2(screenResolutionX / 2 + 120,
screenResolutionY / 2 + 30));

                    enterBack.UpdatePosition(new Vector2(screenResolutionX / 2 - 195,
screenResolutionY / 2 - 47));
                    enterSubmit.UpdatePosition(new Vector2(screenResolutionX / 2 + 75,
screenResolutionY / 2 - 47));



                }
                if(exitOptions.clicked)
                {
```

```
            showOptions = false;
            exitOptions.clicked = false;
        }
    }

    if(gameOver)
    {
        if (!enterNameOn)
        {
            deathQuit.Update(m);
            deathReplay.Update(m);
            deathSubmit.Update(m);
        }
        if (deathQuit.clicked || deathReplay.clicked)
        {
            showPleaseWait = true;
            g = 0;
            time2 = 0;
            deathPlayedOnce = true;

        }
        if (deathSubmit.clicked)
        {
            enterNameOn = true;
            deathSubmit.clicked = false;

        }
        if (enterNameOn)
        {
            enterBack.Update(m);
            enterSubmit.Update(m);

            if (enterSubmit.clicked)
            {
                sendScore(scoreName, Convert.ToInt32(score));
                enterNameOn = false;
            }


            Keys[] pressedKeys;
            pressedKeys = keyBoardState.GetPressedKeys();

            foreach (Keys key in pressedKeys)
            {
                if (oldKeyBoardState.IsKeyUp(key) && key != Keys.Add
                    && key != Keys.Apps
                    && key != Keys.Attn
                    && key != Keys.BrowserBack
                    && key != Keys.BrowserFavorites
                    && key != Keys.BrowserForward
                    && key != Keys.BrowserHome
                    && key != Keys.BrowserRefresh
                    && key != Keys.BrowserSearch
                    && key != Keys.BrowserStop
                    && key != Keys.CapsLock
                    && key != Keys.ChatPadGreen
                    && key != Keys.ChatPadGreen
                    && key != Keys.ChatPadOrange
                    && key != Keys.Crsel
                    && key != Keys.D0
                    && key != Keys.Decimal
                    && key != Keys.Divide
                    && key != Keys.Down
                    && key != Keys.End
```

```
&& key != Keys.Enter
&& key != Keys.EraseEof
&& key != Keys.Escape
&& key != Keys.Execute
&& key != Keys.Exsel
&& key != Keys.F1
&& key != Keys.F10
&& key != Keys.F11
&& key != Keys.F12
&& key != Keys.F13
&& key != Keys.F14
&& key != Keys.F15
&& key != Keys.F16
&& key != Keys.F19
&& key != Keys.F18
&& key != Keys.F2
&& key != Keys.F3
&& key != Keys.F4
&& key != Keys.F5
&& key != Keys.F6
&& key != Keys.F7
&& key != Keys.Help
&& key != Keys.Home
&& key != Keys.Insert
&& key != Keys.Kana
&& key != Keys.Kanji
&& key != Keys.LaunchMail
&& key != Keys.Left
&& key != Keys.LeftAlt
&& key != Keys.LeftControl
&& key != Keys.LeftShift
&& key != Keys.LeftWindows
&& key != Keys.MediaNextTrack
&& key != Keys.MediaPlayPause
&& key != Keys.MediaPreviousTrack
&& key != Keys.MediaStop
&& key != Keys.Multiply
&& key != Keys.None
&& key != Keys.NumLock
&& key != Keys.NumPad0
&& key != Keys.NumPad1
&& key != Keys.NumPad2
&& key != Keys.NumPad3
&& key != Keys.NumPad4
&& key != Keys.NumPad5
&& key != Keys.NumPad6
&& key != Keys.NumPad7
&& key != Keys.NumPad8
&& key != Keys.NumPad9
&& key != Keys.Oem8
&& key != Keys.Pa1
&& key != Keys.PageDown
&& key != Keys.PageUp
&& key != Keys.Pause
&& key != Keys.Play
&& key != Keys.Print
&& key != Keys.PrintScreen
&& key != Keys.ProcessKey
&& key != Keys.Right
&& key != Keys.RightAlt
&& key != Keys.RightControl
&& key != Keys.RightShift
&& key != Keys.RightWindows
&& key != Keys.Scroll
```

```
                            && key != Keys.Select
                            && key != Keys.Separator
                            && key != Keys.Sleep
                            && key != Keys.Subtract
                            && key != Keys.Tab
                            && key != Keys.Up
                            && key != Keys.VolumeDown
                            && key != Keys.VolumeMute
                            && key != Keys.VolumeUp
                            && key != Keys.Zoom
                            && key != Keys.OemTilde
                            )
                        {
                            if(key == Keys.D0 || key == Keys.D1 || key == Keys.D2 || key ==
Keys.D3 || key == Keys.D4 || key == Keys.D5 || key == Keys.D6 ||
                                key == Keys.D7 || key == Keys.D8 || key == Keys.D9)
                                scoreName += key.ToString().ElementAt(1);
                            else
                            if (key == Keys.Back && scoreName.Length != 0) // overflows
                                scoreName = scoreName.Remove(scoreName.Length - 1, 1);
                            else
                                if (key == Keys.Space)
                                    scoreName = scoreName.Insert(scoreName.Length, " ");
                                else if (key != Keys.Back)
                                    scoreName += key.ToString();
                        }
                    }
                }
                if (enterBack.clicked)
                {
                    enterNameOn = false;
                    enterBack.clicked = false;
                }
            }


            if (!gameStarted)
            {
                time += gameTime.ElapsedGameTime.TotalSeconds;
                while (time > 1.0 / 24)
                {
                    if (i >= 1 && i <= 5)
                        showPleaseWait = false;
                    time -= 1.0 / 24;
                    i += 1;



                }
                if (i >= 39)
                {
                    i = 0;
                    return;
                }
            }
            if (gameOver)
            {
                time2 += gameTime.ElapsedGameTime.TotalSeconds;
                while (time2 > 1.0 / 24 && g <= 53)
                {
                    time2 -= 1.0 / 24;
                    g += 1;
                }
```

```
                if(g == 54)
                {
                    if (deathPlayedOnce && soundDeathEerieInstance.State == SoundState.Stopped)
                    {
                        deathPlayedOnce = false;

                        soundDeathEerieInstance.Play();
                    }

                }
            }
        }
        public void Draw()
        {
            Vector2 FontOrigin = scoreFont.MeasureString(score) / 2;


            spriteBatch.Begin(SpriteSortMode.Deferred, BlendState.AlphaBlend);


            if (gameStarted)
            {


                spriteBatch.Draw(vignette, new Rectangle(0, 0, screenResolutionX,
screenResolutionY), Color.White * vignetteAlpha);

                // Draw the string

                spriteBatch.Draw(scoreBoard, new Vector2(screenResolutionX / 2 - 100, 0),
Color.White);
                spriteBatch.DrawString(scoreFont, score, new Vector2(screenResolutionX / 2 - 3,
23), Color.White,
                0, FontOrigin, 2f, SpriteEffects.None, 0.5f);

                if (gameOver)
                {
                    spriteBatch.Draw(deathScreen[g], new Rectangle(0, 0, screenResolutionX,
screenResolutionY), null, Color.White);
                    if(g == 54)
                    {
                        spriteBatch.Draw(deathOptions, new Vector2(screenResolutionX / 2 -
deathOptions.Width / 2, screenResolutionY / 2 - deathOptions.Height / 2), Color.White);
                        spriteBatch.DrawString(scoreFont, score, new Vector2(screenResolutionX /
2 - 7, screenResolutionY / 2), Color.Black,
                            0, FontOrigin, 1.3f, SpriteEffects.None, 0.5f);

                        deathQuit.Draw(scoreFont);
                        deathReplay.Draw(scoreFont);
                        deathSubmit.Draw(scoreFont);

                        if(enterNameOn)
                        {
                            spriteBatch.Draw(enterName, new Vector2(screenResolutionX / 2 -
enterName.Width / 2, screenResolutionY / 2 - enterName.Height / 2 - 50), Color.White);
                            spriteBatch.DrawString(scoreFont, scoreName, new
Vector2(screenResolutionX / 2 + 5, screenResolutionY / 2 - 88), Color.Black,
                            0, FontOrigin, 1.3f, SpriteEffects.None, 0.5f);
                            enterBack.Draw(scoreFont);
                            enterSubmit.Draw(scoreFont);
                        }

                    }
                    if (showPleaseWait)
```

```
                    { }

                }
            }
            else
            {
                spriteBatch.Draw(menu[i], new Rectangle(0, 0, screenResolutionX,
screenResolutionY), null, Color.White);//draw the current (i) image
                play.Draw(scoreFont);
                options.Draw(scoreFont);
                exit.Draw(scoreFont);
                if (showOptions)
                {
                    spriteBatch.Draw(optionWindow, new Vector2(screenResolutionX / 2 - 275,
screenResolutionY / 2 - 137), Color.White);
                    exitOptions.Draw(scoreFont);
                    applyOptions.Draw(scoreFont);
                    spriteBatch.DrawString(scoreFont, resolutionString, new
Vector2(screenResolutionX / 2 + 200, screenResolutionY / 2 - 113), Color.Black,
                0, Vector2.Zero, 1f, SpriteEffects.None, 0.5f);
                    spriteBatch.DrawString(scoreFont, animationString, new
Vector2(screenResolutionX / 2 + 200, screenResolutionY / 2 - 72), Color.Black,
                0, Vector2.Zero, 1f, SpriteEffects.None, 0.5f);
                }


            }

            spriteBatch.End();


        }
    }
}
```

Kvalifikācijas darbs „**3D Spēle un tās mājaslapa**” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: ***Students Pāvels, Jacuks*** _____ **___.05.2015**.

Rekomendēju darbu aizstāvēšanai

Darba vadītājs: ***Datorzinības maģistrs Artūrs Lavrenovs*** _____ **___.05.2015.**

Recenzents: ***Datorzinības maģistrs Pēteris Krastiņš***

Darbs iesniegts 01.06.2015.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: *Darja Solodovņikova*_____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

___.06.2015. prot. Nr. _____

Komisijas sekretārs(-e): _____