|  |  |  |
| --- | --- | --- |
| Image result for univerzitet u novom sadu | UNIVERZITET U NOVOM SADU  **FAKULTET TEHNIČKIH NAUKA U NOVOM SADU** |  |

Nikola Berdić (E2-66/2015)

Naslov

Seminarski rad

- iz predmeta Proces razvoja računarskih igri -

Mentor:

Prof. dr Dragan Ivetić

1. Sadžaj

# **1. Uvod**

Razvoj igara je jedna od mnogobrojnih grana razvoja računarskog softvera. Prve video igre vuku korene još iz šezdesetih godina 20. veka. Iako su tada razvijane po nekolko godina namenski za “mainframe” računare i iako nisu bile dostupne javnosti, predstavljale su prve korake u razvoju interesantnih grafičkih okruženja i veštačke inteligencije [1]. Nakon toga, već sedamdesetih godina, pojavile su se komercijalne video igre koje od tada pa do dana današnjeg, kako za konzole tako i za personalne računare, diktiraju tempo razvoja softverskih i hardverskih komponenti za udobnije korišćenje igara, i njihov brži razvoj. Između pokretačke snage razvoja tehnologija koju računarske igre nose i njihovog postojanja zarad zabave, često se potencira i njihov edukativni smisao. Poseban žanr igara GBL (Game Based Learning [2]) se bavi aplikacijama koje se razvijaju sa ciljem povezivanja željene materije i toka igre, obraćajući pažnju na mogućnost primene stečenih znanja u realnom svetu.

Deca uzrasta do 5 godina su u današnje vreme dosta izložena tehnologiji. Računari, mobilni telefoni i tablet računari su lako dostupni i dosta zastupljeni u porodicama. Ljudska radoznalost teži ka istraživanju i pronalaženju novih stvari, tako i u slučaju dece. Da bi se deca pravilno i dobro razvijala i pored velike izloženosti tehnologiji potrebno je razviti okruženja, igre i aplikacije u skladu sa njihovim uzrastom. Upravo je to motiv razvoja koncepta igre koja bi omogućila pravilan i dobar razvoj dece i njihovo snalaženje u 3D prostoru.

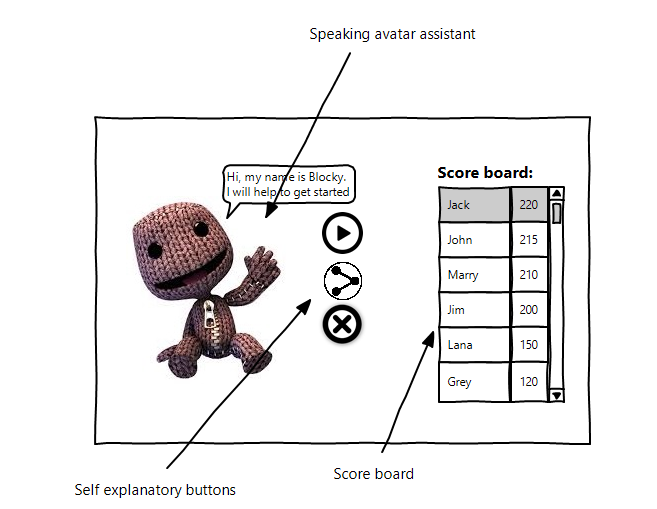
Koncept igre za snalaženje u 3D prostoru bi ojačao percepciju dubine prostora, orientaciju kao i sve ostale prostorne veštine kod dece. Igra bi takođe razvila motoriku za upravljanje današnjih tehnologija, kao što su „drag-and-drop“, „swipe“ i „click“ na touch screen uređajima. User-interfejs bi bio u skladu sa ciljnom grupom. Bilo bi što više simbola umesto reči kao i sličica koji bi indikovale željenu akciju.

Seminarski rad će opisati koncept prethodno objašnjene igre. Takođe biće dat detaljan pregled scenarija i izgleda igre. Tehnologije koje bi bile korišćene za razvoj igre i primenjeni algoritmi. Biće opsiana i ciljana grupa igre, psihološki uticaj takve vrste igre na grupu i beneficije koje bi takav razvoj doneo.

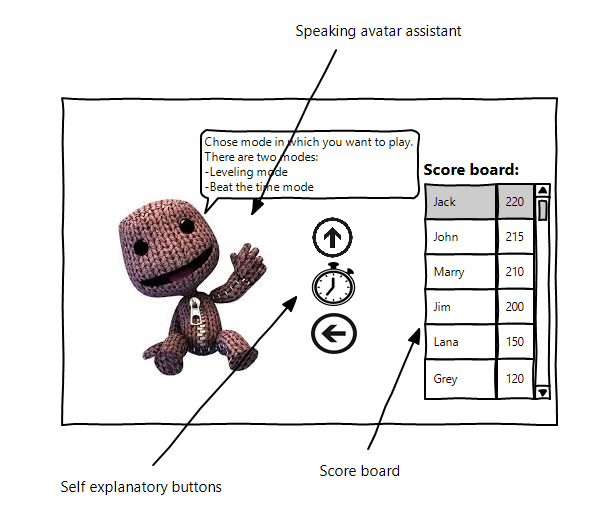
**2. Koncept i scenario igre**

Glavni koncept igre bio bi sklapanje prethodno zadatog 3D meš modela uz pomoć primitivnih vrsti objekata, kao što su kocka, lopta, piramida, valjak i slično. Početni zadati objekat koji bi korisnik morao da reprodukuje bio bi upravo jedan od prethodno navedenih primitivnih objekata. Rešavanjem svakog prethodnog zadatog zadatka povećavala bi se kompleksnost zadatog 3D meš modela koji je potrebno reprodukovati. Ovakav koncept igre bi naterao razvija korisnikove prostorne veštine i na taj način bi konstantno napredovao. Da bi koncept igre i njen interfejs bio prikazan korišćen je alat WireframeSketcher 4.7.0 [3]. Alat služi za skiciranje i kreiranje koncepta interfejsa aplikacija.

Na slici 1 je grub pikaz interfejsa prethodno opisane igre. Interfejs bi bio veoma user-friendly. Sadržao bi slikovite dugmiće koji bi bili samoobjašnjivi. Bila bi prikazana i trenutna tabela sa njaboljijm rezultatima. Na taj način bi korisnik težio da ostvari što bolji i zapaženiji rezultat. Takođe kroz celu igru prožimao bi se avatar asistent koji bi ima govorne sposobnosti i na taj način bi olakšao korisniku korišćenje aplikacije. Korisnik ne bi morao da čita već bi mu virtuelni asistent objasnio šta je potrebno da učini. Kao i svaka igra i ova igra bi imala tutorijal u kojem bi uz pomoć virtuelnog asistenta korisnik stekao potrebno znanje za korišćenje aplikacije.

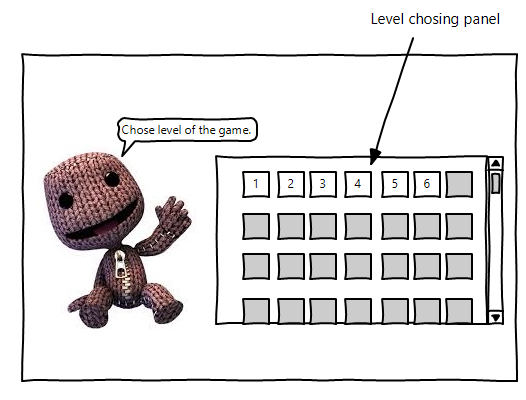
Slika 1. Grub prikaz interfejsa igre.

Igra bi imala dva moda koje bi korisnik odabrao pre početka igre (slika 2). Jedan mod bi bio pelaženje nivoa, dok bi drugi bio vremenski ograničen. Mod u kome bi korisnik prelazi nivoe je jedan od najzastupljenijih koncepta igara današnjice. Nakon svakog uspešnog zadatka korisnika bi čekao malo teži zadatak. Na taj način bi korisnik inkrementalno razvijao svoje sposobnosti i veštine. U slučaju ove igre, prostorne veštine.

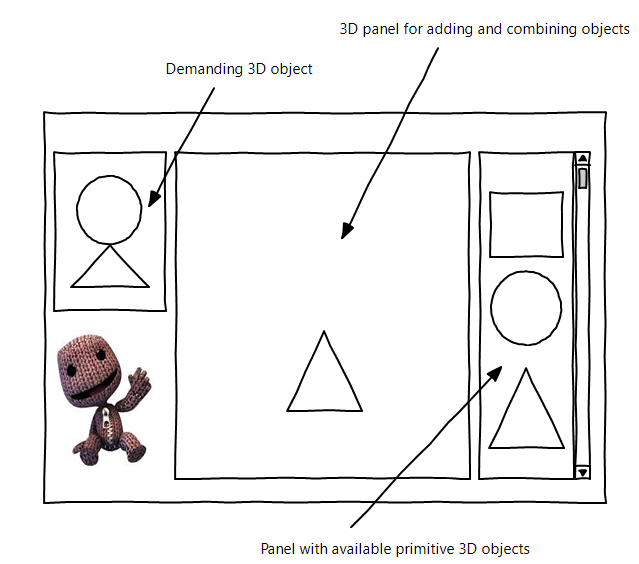


Slika 2. Grub prikaz odabira moda igre u kojem bi korisnik igrao

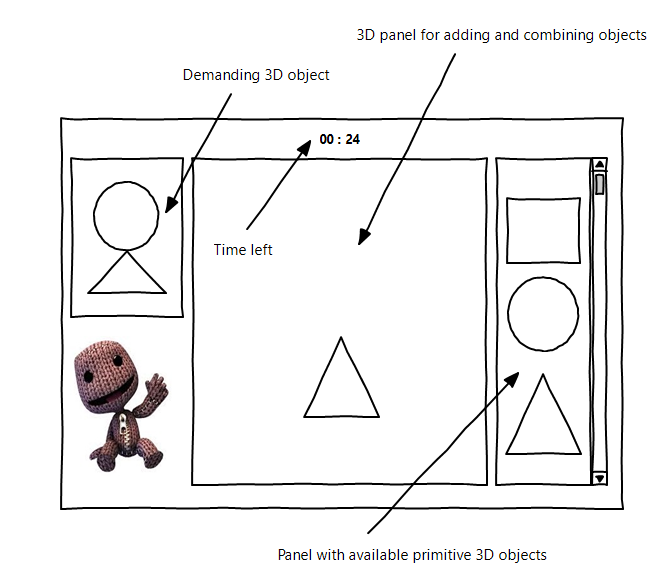
Nakon što bi korisnik odabrao mod za prelaženje nivoa otvorio bi mu sledeći ekran na kome bi mogao da odabere nivo, slika 3. Svaki pređeni nivo bio bi dostupan korisniku za ponovni prelazak. Na taj način bi korisnik mogao iznova i iznova da testira svoju brzinu i svoj napredak nakon dužeg perioda igranja. Dok svaki nepređeni nivo ne bi bio dostupan. Nivoe bi korisnik otključavao tako što bi prešao prethodni nivo. Na taj način bi igra „terala“ korisnika da stiče potrebno iskustvo i da razrađuje svoje veštine kako bi mogao da pokuša da pređe željeni nivo.

Slika 3. Grub prikaz interfejsa za odabir nivoa koji bi korisnik igaro.

Vremenski ograničen mod bi značio da korisnik ima određeni period vremena da reprodukuje što više ponuđenih nasumično odabranih figura. Suma težina figura koje je korisnik uradio za fiksno određeno vreme. Oba načina igre bi imala istu svrhu s tim i vrlo sličan interfejs (slika 4). Prozor na kome bi služio za glavni deo igre imao bi tri panela. Prvi panel i koji bi se nalazio na levoj strani prozora bio bi panel u kome bi bio prikazan željeni 3D meš model. Kreiran na taj načina da je moguće reprodukovati ga sa primitivnim 3D objektina. U sredini bi bio panel u kome bi korisnik dodavao primitivne 3D objekte i manipulisao njima. Pomerao bi ih, rotirao i skalirao tako da bi se što više uklopili u željeni 3D meš model. Treći panel koji je pozicioniran na desnoj strani prozora bio bi panel u kome bi se nalazili svi primitivni 3D objekti, kao što su sfera, piramida, kupa, valjak, kocka itd.

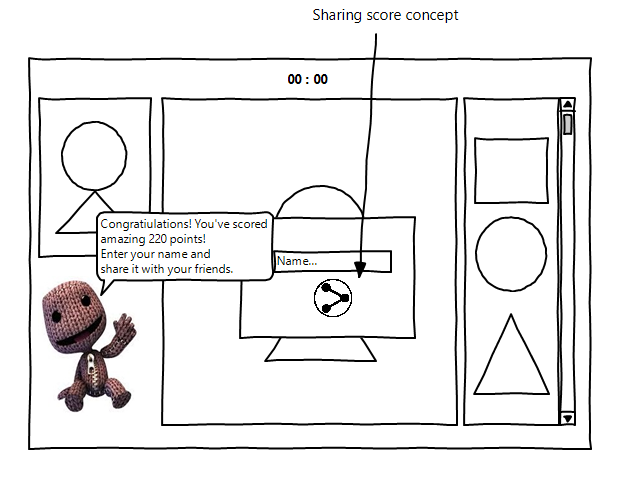
Slika 4. Grub prikaz interfejsa u kome bi korisnik kreirao željene objekte u modu prelazka nivoa

Interfejs u kome bi korisnik igrao igru u vremenski ograničenom modu bio bi vrlo malo drugačiji (slika 5). Za razliku od moda u kome korisnik prelazi nivoe, interfejs u kome korisnik ima ograničeno vreme sadrži i indikator preostalog vremena.



Slika 5. Grub prikaz interfejsa u kome bi korisnik kreirao željene objekte u modu ograničenog vremena

Nakon što bi korisnik uspešno završio jedan od dva moguća moda mogao bi da deli svoj rezultat sa ostalim prijateljima i takmičarima koji igraju igru, slika 6. Taj koncept i mogućnost bi ga podstakla da se upoređuje i uvek teži ka tome da ostvari što bolji rezultat.



Slika 6. Grub prikaz interfejsa za deljenje rezultata

**3. Razvojna tehnologija i algoritmi**

Tehnologija za razvoj 3D igara je u velikoj ekspanziji. Da bi se započeo razvoj kompleksne 3D igre potrebno je istražiti i odabrati najbolji game engine. Game engine predstavlja skup osnovnih funkcionalnosti koje su potrebne za razvoj 3D igara. U funkcionalnosti spadaju rendering motor za 2D i 3D grafiku, fizički motor ili motor kolizije, zvuk, skripting, animaciju, veštačku inteligenciju, upravljanje memorijom itd. Jedni od najpopularnijih game engine-a su: Unreal Engine 4, Unreal Engine 3, Unity 3D, Torque 3D...[4]. U slučaju igre koja ima koncept opisanog u prethodnom poglavlju nije potrebno korišćenje game engine-a, razlog je zato što igra zahteva jednostavne funkcije koje je moguće izvući iz manje zahtevnih programa.

Prethodno opisani koncept igre moguće je razviti u Python[5] objektno orijentisanom programskom jeziku verzije koja ima podršku Pygame[6] interface-a. Pygame interfejs sadrži dosta funkcionalnosti koje su vrlo lake za korišćenje i ne toliko zahtevne kao prethodno navedeni game engine-i. Takođe, algoritam koji određuje stepen poklapanja zadatog 3D meš modela sa modelom koji korisnik kreira ima implementaciju u Python programskom jeziku. Naravno, ovo ne treba da znači da je ovo najbolji izbor. Sve zavisi od programera i onoga u čemu on misli da bi se najbolje snašao prilikom razvoja ovakve igre.

**3.1. Python i Pygame**

Python objektno orijentisani programski jezik je jedan od najrasprostranjenijih i najpopularnijih objektno orijentisani programskih jezika u svetu [7]. Dizajniran je tako da kod napisan u Python-u bude što čitljiviji. Sintaksa je takva da dozvoljava programerima da prikažu svoj koncept u što manje linija u odnosu na Javu ili C++.Takođe Python prevodioci su dostupni za mnoge operativne sisteme, što dozvoljava da se Python koristi na različitim operativnim sistemima. Korišćenjem alata razvijene od strane treće strane kao što su Py2exe ili Pyinstaller, Python kod je moguće spakovati u jedan samostalan izvršni fajl za neke od najpopularnijih operativnih sistema.

Pygame je skup Python modula dizajniranih za pisanje igrica. Pygame dodaje funkcionalnosti na već odlični SDL[8]. Na ovaj način programer može da razvije igre i multimedijalne programe u Python-u. Pygame, kao i Python, je moguće koristiti na skoro svim operativnim sistemima. Jedni od operativnih sistema koje podržava su Linux, Windows, BeOS, MacOS, NetBSD, Solaris itd. Jedna od glavnih karakteristika Pygame i razloga zašto je toliko popularan za početak razvijanja ne tako složenih igri je koncept kojim su se vodili prilikom njegovog kreiranja, a to je da olakšaju što više moguće razvoj i da ga učine zabavnim. Takođe, da bi funkcionisao Pygame ne zahteva ekskluzivno instaliran OpenGL. Moguće je korišćenje više jezgara procesora veoma lako i samim tim igre bi bile brže i ažurnije. Koliko je jednostavno kreirati nešto u Pygame-u prikazano je u kodu 1. U samo par linija jednostavnih i vrlo čitljivih linija koda kreirana je funkcionalnost „drag and drop“ za krug. Kreiranje iste te funkcionalnosti, ali za 3D objekte, zahtevao bi malo veći napor i malo više uloženog vremena.

Kod 1. Prikaz jednostavnog korišćenja Pygame modula u Python-u za „drag and drop“ efekat kruga.

|  |
| --- |
| from pygame import \*  screen=display.set\_mode((600,400)) # postavljanje velicine prozora  running=True  FirstFrameDown=True  while running:  for evnt in event.get(): #preuzimanja svih event-ova  if evnt.type == QUIT: # ukoliko je zahtevani event izlazak, izaci  running=False    x,y=mouse.get\_pos() # uzimanje pozicije misa  b=mouse.get\_pressed() # uzimanje pritiska misa  if evnt.type == MOUSEBUTTONDOWN:  if evnt.button == 1:  if FirstFrameDown == True: # proveravanje da li je prvi frejm  screencopy = screen.copy() #ukoliko jeste napravi novi objekat celog prozora  FirstFrameDown=False # i postavi flag na false  if evnt.type == MOUSEBUTTONUP: # proveri da li je pusten pritisak  FirstFrameDown=True # resetuj flag za prvi frejm  if b[0] == 1:  screen.blit(screencopy,(0,0)) #ukoliko ovoga ne bi bilo krugovi bi se konstanto  #crtali prilikom pritiska, stvarajuci efekat npr cetkice u paint-u  draw.circle(screen,(255,0,0),(x,y),20) # crtaj na poziciji misa crveni krug  display.flip()  quit() |

**3.2. Algoritam za određivanje stepena poklapanja**

Prethodno navedene tehnologije mogu se iskoristit kao razvojno okruženje ovakve igre, implementaciju osnovnih funkcija za interakciju korisnika sa aplikacijom kao i kreiranje interfejsa za lakše i bolje navigiranje kroz samu igru. Da bi koncept igre bio u potpunosti funkcionalan potrebno je implementirati i algoritam koji bi određivao stepen poklapanja zadatog meš modela i konstruisanog meš modela od strane korisnika. Razvijeni algoritam je iskoristio već razvijenu metodu poznatiju kao Non Rigid Registration [9].

Glavna ideja metode [9] je da se meri sličnost između dva konačna seta tačaka uzimajući u obzir njihovu kontinualnu aproksimaciju. Zbog toga moguće je povezati set tačaka sa funkcijom verovatnoće gustine. Uzmimo u obzir set tačaka kao kolekciju Dirac Delta funkcija, normalno je predpostaviti da je konačana modela mešavina predstavljena kao set tačaka. Kao najčešće korišćena model mešavina, Gaussianova model mešavina kao konveksna kombinacija Gaussianovih komponenti gustine ϕ(**x**|μi,Σi) , gde je μi srednji vektor, a Σi matrica kovarianse. Funkcija verovatnoće gustine je data formulom 1. Gde wi predstavlja težinu komponenti. Ukoliko je broj komponenti, *k,* veliki onda bilo koja gustina može biti određena ovim modelom. U radu [9] iskorišćena je Gaussianova model mešavina da predstavi eksplicitno set tačaka. U prostijim podešavanjima, broj komponenti je broj setova tačaka.

Formula 1. Funkcija verovatnoće gustine

Algoritam koji primenjuje [9] i koji bi bio pogodan za određivanje stepena poklapanja meš modela prikazan je u kodu 2. Kod je implementiran u Python-u i samim tim bi mogao lako da se koristi ukoliko bi se igra interfejs razvio Pygame-u. Pošto korisnik ima mogućnost da vrši rotaciju, translaciju i skaliranje primitivnih objekata kako bi dobio željeni i traženi objekat, potrebno je na početku algoritma tačke izvajanog meš modela skalirati na veličinu traženog meše modela, jer ukoliko se to ne bi uradilo onda iako je model možda isti kao i traženi njegove tačke se ne bi poklapale. Nakon uspešnog skaliranja izvajanog modela pomoću metode [9] vrši se Non Rigid Registration nad tačkama idealnog modela i izvajanog modela. Funkcija vraća Vraća unete tačke model i scenu, kao i tačke koje su generisane nakon funkcije i obavljene metode. Zatim, da bi se odredila udaljenost i da li je poklapanje modela ispravno, potebno je odrediti udaljenost izmedju tačke izvajanog modela i tačke generisane nakon non-rigid-registration metode. Ukoliko je se modeli dobro poklapaju međusobne tačke bi bile na minimalnoj udaljenosti jedna od druge. Pak, ukoliko se modeli ne uklapaju dovoljno dobro onda bi postojale tačke koje u svojoj blizini nemaju drugu tačku i samim tim modeli nisu dobro poklopljeni. U kodu određivanje tačaka u blizini odrađen je pomoću objekta BallTree. Konstruktor objekta uzima kao ulazni argument skup tačaka. Funkcijom query\_radius(point, radius) prolazi se kroz skup tačaka definisanih u konstruktoru i za svaku tačku se ispituje da li se nalazi u sferi radijusa radius. Ukoliko se tačka nalazi unutar sfere onda se ta tačka dodaje u niz tačaka koje se nalaze u sferi koje se na kraju funkcije vraćaju kao povratna vrednost funkcije.

Kod 2. Implementacija algoritma za određivanje stepena poklapanja dva meš modela u Python-u

|  |
| --- |
| def get\_score\_for\_ideal\_points(model\_points, ideal\_points, IDEAL\_RADIUS, IDEAL\_DISTANCE):  #generisi normalizovane tacke modela  points = get\_normalized\_points(model\_points,IDEAL\_RADIUS)  #odradi non rigid registration nad normalizovanim modelom i idealnim  model,scene,after\_tps = cca.non\_rigid\_registration(points, ideal\_points)  distances\_array = []  ballTree = BallTree(after\_tps)    radius = 2\*IDEAL\_DISTANCE/3  not\_functional = []  i = 0  for point in ideal\_points:  #pomocu ballTree objekta preuzmi sve tacke koje se nalaze u sferi radiusa radius u odnosu na tacku point  ind = ballTree.query\_radius(point, radius)  if len(ind[0]) == 1:  #ukoliko postoji neka tacka unutar sfere dodaj distancu izmedju tacaka u niz distanci  distances\_array.append(np.linalg.norm(point - after\_tps[ind[0][0]]))  else:  #ukoliko ne postoji nijedna tacka u okolini dodaj veliku distancu u niz kako bi imalo veliki uticaj na srednju vrednost  distances\_array.append(1000)    return np.mean(distances\_array) |

**4. Ciljna grupa**

Ceo koncept igre i njen scenario se svodi na to da korisnik na jednostavne načine pristupi i koristi igru koja bi njemu razvila prostorne veštine, orientaciju i bolju percepciju dubine prostora.

**6. Literatura**

[1] Video game development, http://en.wikipedia.org/wiki/Video\_game\_development

[2] Game based learning, http://en.wikipedia.org/wiki/Game\_based\_learning

[3] WireframeSketcher, <http://wireframesketcher.com/>

[4] Game Engines , <http://www.worldofleveldesign.com/categories/level_design_tutorials/recommended-game-engines.php>

[5] Python, <https://www.python.org/>

[6] Pygame, <http://www.pygame.org/hifi.html>

[7] TIOBE index, <http://www.tiobe.com/tiobe-index/>

[8] Simple Directmedia Layer (SDL), <http://www.libsdl.org/>

[9] Bing Jian and Baba C. Vemuri - A Robust Algorithm for Point Set Registration Using Mixture of Gaussians