Razvoj igre za ekran na dodir

Kolendarić, Marko

SVEUČILIŠTE U ZAGREBU

GRAFIČKI FAKULTET

ZAVRŠNI RAD

Marko Kolendarić

Smjer: Tehničko tehnološki

ZAVRŠNI RAD

RAZVOJ IGRE ZA EKRAN NA DODIR

Mentor:

Dr.sc. Klaudio Pap

Student:

Marko Kolendarić

Zagreb, 2015.

SAŽETAK:

Cilj ovog rada je prikazivanje postupka izrade jedne igre za uređaje na dodir, odnosno današnje mobilne uređaje poput mobitela, tableta i sl. Prikazom postojećih tehnologija koje se koriste u tom polju, od početaka, pa preko razvoja do današnjih izvedbi, bit će objašnjen cjelokupni postupak, kako izrade, tako i ostalih funkcija koje se javljaju u poljima izrade igara, kako onih za ekrane na dodir, tako i općenito. U eksperimentalnom dijelu ovog rada bit će detaljno analiziran primjer jedne takve igre kako bi se na što jednostavniji način demonstrirali svi alati te sustavi koji se danas koriste. Na tom primjeru igre, bit će vidljiva jednostavnost te lakoća izvedbe takvih projekata u današnje doba, ali i kompleksnost koja se javlja kod različitih problema koji se mogu javiti pri izvedbi iste. Rješavanjem i izvedbom jednog takvog procesa obrađene su sve tematske cijeline koje su potrebne za rad u toj struci, te kao krajnji rezultat trebaju stvoriti uvid u polje programiranja i igara koja su u jednu ruku budućnost grafičke struke.

KLJUČNE RIJEČI: igra, programiranje, android, ekrani na dodir

SADRŽAJ

1.UVOD....1
2. TEORETSKI DIO....1
2.1. Povijest....1
2.1.1.Prva generacija....1
2.1.2. Druga generacija....3
2.1.4. Četvrta generacija....7
2.1.5. Peta generacija....7
2.1.6. Šesta generacija....8
2.1.7. Sedma generacija....9
2.1.8. Osma generacija....11
2.2. Razvoj ekrana na dodir....12
2.3. Razvojni proces....13
2.3.1. Pred-produkcija....14
2.3.2. Produkcija....15
2.3.3. Post-produkcija....17
2.4. Programski alati....17
2.4.1. Objektno orijentirano programiranje....18
2.4.2. Programski jezici....18
2.4.3. API i knjižnice....19
2.4.4. IDE Sučelja....20
2.5. Distribucija finalnog proizvoda - Marketing....20
2.5.1. Steam....21
2.5.2. Google Play....22
2.5.3. I-Tunes....22
2.5.4.Social networks....22
3. PRAKTIČNI DIO....24
3.1. Razvoj igre....24
3.2. Programski kod igre....24
4. ZAKLJUČAK....40
5. LITERATURA....41

1.UVOD

Obradom problema koji nastaju prilikom izrade računalnih i mobilnih igara, pa tako i igara za ekrane na dodir, može se stvoriti predodžba te detaljnije objasniti postupak izrade takvih igara, tehnologija i procesa koji se koriste u tom polju, te na kraju, vizualnim prikazom finalnog rješenja povezati cijeline koje će biti obrađene prethodno u radu.

2. TEORETSKI DIO

2.1. Povijest

2.1.1.Prva generacija

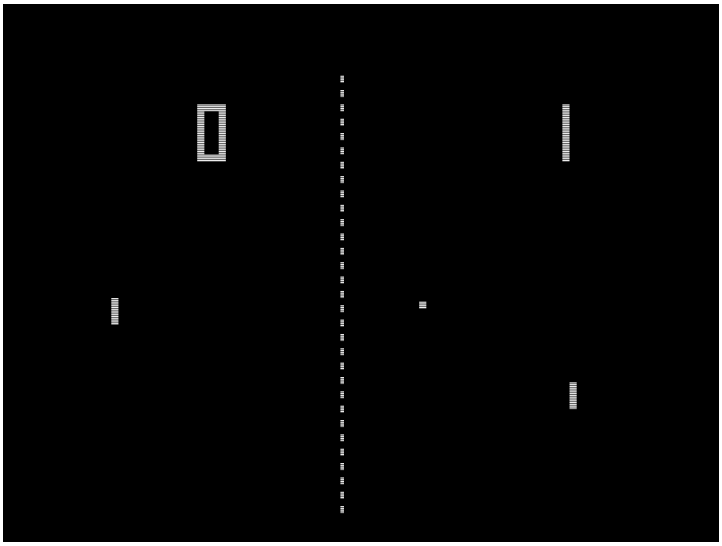
Povijest video igrica počinje 1940-ih godina kada su Thomas T. Goldsmith, Jr. i Estle Ray Mann podnijeli zahtjev Uredu za patente SAD-a za inovaciju koji su opisali kao uređaj za zabavu od katodne crvene cijevi. Video igranje neće doseći najveću popularnost do 70-ih i 80-ih godina, kada su arkadne igre, igraće konzole i kućna računala upoznati sa javnošću. Od tada, video igranje je postalo popularna vrsta zabave. [1]

1959. – 1961., napravljena je kolekcija interaktivnih grafičkih programa na uređaju TX-0 na MIT-u: Miš u labirintu, HAX i Tic-Tac-Toe.

Prva video igrica je bila napravljena na uređaju za zabavu od katodne crvene cijevi te se radila o projektilnom simulatoru koji je inspiriran radarima iz Drugog svjetskog rata. Koristio je analogne sklopove za kontroliranje CRT zrake i pozicioniranje točke na ekranu. Nakon nje su uslijedile igre šaha, kružića i križića, tenisa za dvoje, miš u labirintu, svemirski rat. [1]

1971. godine napravljena je igrica *Galaxy Game*, te je bila prva igrica koja se mogla igrat ako se u aparat ubacio novčić.

Pong, igrica izdana 1972., je prva koja je imala veći uspjeh nego prijašnje. Bazirana je na stolnom tenisu: loptica je servirana sa centra terena te igrači moraju pomicati svoje rekete kako bi udarili lopticu.



Slika 1. Pong igrica

[http://en.wikipedia.org/wiki/Pong]

Gun fight je igrica izdana 1975. godine koja je prva prikazivala nasilje, likove te borbu čovjek na čovjeka. Te je kotrolirana pomoću dvije palice. Također je prva igra koja je koristila mikroprocesor Intel 8080. [1]

Razvoj konzola je počeo 1966. dok je prvi prototip dovršen 1968.te se zvao *Magnavox Odyssey*. Koristio je spremnike koji su se sastojali od skakača koji su (ne)omogućavali razne prekidače unutar te jedinice. To je omogućavalo igranje više različitih igara koristeći isti sistem. Ostale konzole koje su se pojavile od 1972. do 1977. su: [2]

Tablica 1. Prva generacija konzola

Magnavox Odyssey	Serija Magnavox Odyssey	Atari/Sears Telegames Pong	Coleco Telstar	Nintendo Color TV Game
1966	1975	1975	1976	1977
				

[http://en.wikipedia.org/wiki/History_of_video_game_consoles_(first_generation)]

2.1.2. Druga generacija

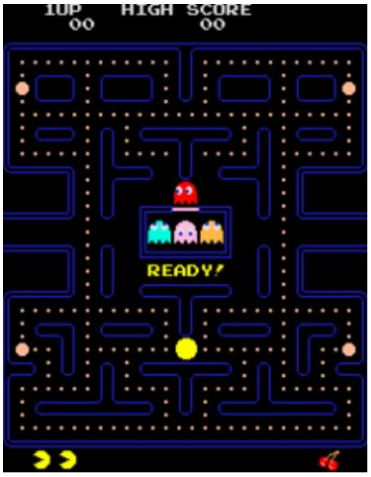
Prva generacija konzola je napravljena tako da je služila samo jednoj igri. Nije bilo software-a nego samo hardware-a, pa je to predstavljalo problem developerima pošto su kupci morali kupiti novi uređaj da ga priključe na svoj televizor kako bi igrali razne igre. Do sredine 70-ih godina video igrice su imale spremnike. To je počelo 1976. puštanjem u javnost sistema VES (Fairchild Video Entertainment System). Programi su se pržili na ROM čipove koji su se nalazili u spremnicima koji su se mogli staviti u konzolu. Tako su korisnici mogli ima jako puno spremnika sa igricama a jednu konzolu. [2]

Tablica 2. Druga generacija konzola

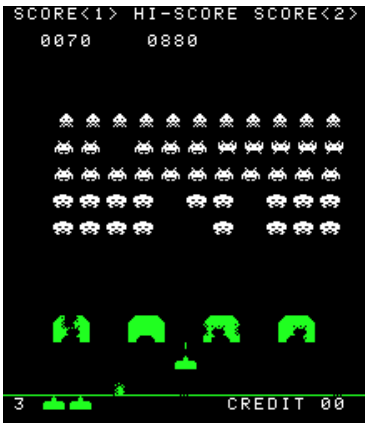
Fairchild Channel F	Atari 2600	Magnavox Odyssey²	Intellivision	Atari 5200
1976	1977	1978	1979	1982
				

[http://en.wikipedia.org/wiki/History_of_video_game_consoles_(second_generation)]

Zlatno doba arkadnih video igara je počelo puštanjem igre Sapce Invaders, čiji je uspjeh potaknuo mnoge da se uključe u proizvodnju igara. Zbog te igre su se uređaji za igranje postavljali u trgovačkim centrima, restoranima i ostalim malim trgovinama. Igra je postala česta tema novina i vijesti na televiziji. Igre u boji su također postale popularne 1979. i 1980. dolaskom Pac-Man-a. [1]



Slika 2. Pac-Man [http://en.wikipedia.org/wiki/Pac-Man]



Slika 3. Space Invaders [http://en.wikipedia.org/wiki/Space_Invaders]

Računalne igre se ne igraju na posebnim konzolama ili uređajima, nego na osobnom računalu. Mnogo igara je napravljeno za igranje na računalu. Prvo su to bile igre po uzoru na televizijske serije i filmove, npr. Star Trek, te kasnije klonovi popularnih arkadnih igara kao Space invaders, Pac-Man, Donky Kong. Distribuirane se disketama ili kazetama poštom. [1]

1980. su donjele mnogo novih i različitih žanrova igara. Neke od njih su:

- Akcijsko advanturističke igrice: *The Legend of Zelda* u kojoj su kombinirani različiti žanrovi kako bi se napravio hibrid koji uključuje puzle, advanture, akcije, monetarni sistem. Ova igra je bila jedan od ranijih primjera otvorenog svijeta, gdje lik kojim korisnik igra može slobodno šetati virtualnim svijetom te mu je dana sloboda izbora. [1]

Slika 4. The Legend of Zelda
[http://en.wikipedia.org/wiki/The_Legend_of_Zelda]

- Akcijske role playing games: *Dragon Slayer II: Xanadu* koja je smatrana prvom potpunom RPG igra sa likom koji počinje ogromno istraživanje, te uključuje akcijske borbe. [1]
- Advanturističke igre: *Zork*, tekstualna igra koja je tako napravljena zato jer je tadašnjim osobnim računalima nedostajalo grafičkih sposobnosti. U sučelje su se pisale naredbe. [1]
- Borbene igre: *Karate Champ*, borbena igra jedan na jedan, gdje se korisnik može boriti sa različitim protivnicima. [1]
- Labirintske igre: *Pac-Man*, gdje je glavni lik u igrici prvi lik koji se popularizirao neovisno o samoj igrici. [1]
- Trkaće igre: *Turbo*, prva trkaća igrica sa pogledom treće osobe. [1]

Nakon uspjeha Apple II i Commodore PET, došla je serija novih i jefitnijih računala koji su pomogli ubrzati tržište kućnih računala i igrica. Nakon što su konzole osjetile pad 1984. računala su nudila jednaku sposobnost igranja. [1] Neka od računala koja su na tržište došla 1980-ih : Commodore 64, Sinclair ZX80 i Atari 8-bitna serija.

2.1.3. Treća generacija

Američko tržište konzolama za video igre je 1985. oživilo sa objavljivanjem Nintendo-ve 8-bitne konzole – Nintendo Entertainment System (NES). NES je je odmah ostvario uspjeh, dominirajući Sjevernom Amerikom i Japanom. U novim konzolama *joystick*, palice, tastature su zamjenjni *gamepad*-ima. [1]

Slika 5. Nintendo Entertainment System
http://en.wikipedia.org/wiki/Nintendo_Entertainment_System]

Handheld igraće konzole su mali mobilni uređaji za igranje igara koje su često bile minijatnurna verzija prave video igre. Obično su mogle igrati jednu igru.

Nintendo je prvi objavio ručnu konzolu Game Boy 1989., 10 godina nakon što je Microvision objavio svoju nesretnu verziju konzole. Popularna igrica *Tetris* je bila već uključena u konzolu kada bi je korisnici kupili. Prvu ručnu konzolu sa ekranom u boji je objavio Atari Lynx. [1]

2.1.4. Četvrta generacija

Razdoblje 4. generacije konzola je obilježena suparništvom između konzola Nintendo-a i Sega-e:

Slika 6. Sega Genesis
[http://en.wikipedia.org/wiki/Sega_Genesis]

Nintendo je uspio preuzeti svjetsko tržište dok je Sega započela novu franšizu, *Sonic the Hedgehog*, koji bi se natjecao za prvo mjesto za Nintendo-vom serijom Mario. [2]

2.1.5. Peta generacija

Atari je ponovno ušao na tržište konzola sa *Atari Jaguar* konzolom. No konzola je propala pošto nije bilo nikakvog profita te nekvalitetnih igrica. 1994. objavljene su tri nove konzole. Sega Saturn, Sony PlayStation i PC-FX. PlayStation je nadmašio svoje konkurente osim NES-a kojeg su još uvijek mnogi podržavali. [2]

Slika 7. Sony PlayStation
[http://en.wikipedia.org/wiki/Sony_PlayStation]

Nintendo je 1995 izdao Virtual Boy koji je omogućavao 3D percepcijuali nije postigao visoku prodaju zbog monokromatskog ekrana pa je maknut sa tržišta. Sljedeće godine je Nintendo izdao 64-bitnu konzolu, Nintendo 64. [2]

Slika 8. Nintendo 64
[http://en.wikipedia.org/wiki/Nintendo_64]

2.1.6. Šesta generacija

Platforme koje je uključivala šesta generacija konzola su: Sony Playstation 2, Sega Dreamcast, Nintendo GameCube i Microsoft Xbox. Dreamcast je bila prva konzola koja je uključivala ugrašeni modem za internet podršku i igranje. PlayStation 2 je uključivao DVD igre sa kapcitetom od 4.7 GB, povećanim procesorom, ugrađenom konekcijom za četiri igrača, sposobnošću puštanja glazbe i filmova. GameCube je bio prva konzola bazirana na optičkom disku. Microsoft je ušao u tržište konzolama sa svojim Xbox-om. Konzola je imala veliki dio računalne tehnologije što je omogućilo igranje računalnih igrara na Xbox-u. Kratko nakon što je Xbox izašao, napravljena je igra *Halo: Combat Evolved* koja postala najveći uspjeh Xbox-a. 2001. Grand Theft Auto III je objavljen te je imao mnogo uspjeha kritički i komercijalno. [2]

Slika 9. GameCube
[http://en.wikipedia.org/wiki/GameCube]

Slika 10. PlayStation 2
[http://en.wikipedia.org/wiki/PlayStation_2]

Kako se pristupačna internet veza sve više širila, mnogi izdavači su se okrenuli igranju preko interneta. Neke od prvih takvih igara su bile: *World of Warcraft, EverQuest i Ultima Online*. Izdavale su se posebno za osobno računalo. Prva konzola na kojoj se moglo igrati preko interneta sa drugim igračima je bio Dreamcast, na kojem se igrala igra *Phantasy Star Online*. Nakon nje je došla igra *Final Fantasy XI* za PlayStation 2. Microsoft je imao svoj servis za igranje preko interneta – Xbox Live. [1]

2.1.7. Sedma generacija

Prve konzole sedme generacije su bile ručne mobilne konzole. Prva takva konzola je bla Nintendo DS te je ubrzo nakon nje uslijedio PlayStation Portable (PSP). Dok je PSP imao veću snagu i grafiku, Nintendo je ostao privržen svojoj običnoj grafici ali novijem dizajnu sučelja. DS je imao dva ekrana od kojih je jedan bio osjetljiv na dodir koji se učinio popularnim kod korisnika a najviše kod djece. [2]

Slika 11. Nintendo DS
[http://en.wikipedia.org/wiki/Nintendo_DS]

Kod uobičajenih konzola, Microsoft je prvi izdao Xbox 360, te ga je Sony pratio s Playstation 3. Pošto je PlayStation 3 podržavao čitanje Blu-ray diskova te je imao Wi-Fi, bio je i najskuplja konzola do tada. Nakon Sony-jeva izdavanja PlayStation-a 3, nintendo je izdao Wii konzolu te ga je ta konzola vratila na tržište. Wii je imao manje tehničke karakteristike nego Xbox 360 i PlayStation 3, imao je palicu bez kabela koja je imitirala pokrete tijela. Pomoću njega se mogao igrati tenis, golf i bejzbol, te su ga korisnici zbog toga i prihvatili, pošto nije imao prave igre sa pričom i likovima. [2]

Slika 12. Wii
[http://en.wikipedia.org/wiki/Wii]

Slika 13. PlayStation 3
[http://en.wikipedia.org/wiki/PlayStation_3]

2.1.8. Osma generacija

2010. Microsoft je izdao Kinect koji je koristio senzor i uređaj s dvije kamere kako bi uhvatio pokret, tako postavši prvi uređaj koji može pratiti igrača u 3D prostoru bez upotrebe joystick-a (palice). [2]

Slika 14. Kinect
[http://en.wikipedia.org/wiki/Kinect]

Prva ručna konzola osme generacije je bio Nintendo 3DS koji ima 3D grafiku, tri kamere, senzor pokreta, žiroskop te *Slide Pad* koji omogućava 360 DEGRES ANALOG INPUT. Nintendo je unaprijedio 3DS konzolu te je izdao 3DS XL koja ima 90% veći ekran i veći život baterije. [2]

Slika 15. Nintendo 3DS
[http://en.wikipedia.org/wiki/Nintendo_3DS]

2011. Sony je najavio PlayStation Vita ručnu konzolu sa OLED ekranom osjetljivim na dodir na više točaka, dvije analogne palice, 3G i Wi-Fi konekciju. [2]

Slika 16. PlayStation Vita
[http://en.wikipedia.org/wiki/PlayStation_Vita]

Nintendo je 2012. izdao konzolu Wii U koja je formalno započela razdoblje osme generacije konzola. Wii U ima HD grafiku, ekran osjetljiv na dodir koji je ugrađen u poseban kontroler za dodane informacije i aktivnosti. Takav kontroler omogućava igranje igara bez televizora. [2]

2.2. Razvoj ekrana na dodir

Ekran na dodir izumio je E.A. Johnson 1965. godine. Takav ekran omogućje korisniku izravnu komunikaciju s onim što je prikazano na uređaju. Današnji ekran na dodir je prisutan na većini igraćih konzola, tabletima, mobitelima, laptopima te čanulanim monitorima. [3]

Ekran na dodir radi na 3 osnovna principa: otporni, kapacitativni i površinski. Otporni sistem se sastoji od normalnog stakla koji je prekriven vodljivim otpornim metalnim slojem. Kroz ta dva sloja se provodi električna struja dok je ekran uključen. Kada korisnik takne ekran, ta dva sloja se spoje na mjestu koje je dodirnuo korisnik. Promjena u električnom polju je zabilježena te se računaju koordinate gdje je došlo do promjene. Nakon što su kooridnate izračunate, posebni program ih pretvara u nešto što perativni sistem može razumijeti, kao što program za miš na računalu pretvara pokrete miša u klikanje ili pomicanje. [3]

Slika 17. Primjer ekrana na dodir
[http://www.citelighter.com/technology/technology/knowledgecards/touch-screen-technology]

2.3. Razvojni proces

Razvoj igre je programski razvojni proces kao što je video igra program umjetnički dijelom (crteži, animacije) i zvukovima. Igre sa slabim razvojem metodologije imaju veće šanse da pređu budžet i vrijeme do kojeg su trebale biti napravljene, te mogu imati jako puno grešaka. [4]

2.3.1. Pred-produkcija

Pred-produkcija je planiranje projekta fokusirano na ideju i koncept razvoja i produkcije i osnovnog dizajna. Cilj konceptualno razvoja je napraviti jasnu i jednostavnu dokumentaciju koja opisuje sve zadatke. Konceptualna dokumentacija može biti razdvojena u tri dijela: visoki koncept, PITCH i koncept. [4]

Dizajn same igrenaravno počinje s idejom koja spada u neke od žanrova igara. Dizajneri često ekperimentiraju sa žanrovima pa igra spada u više njih. Dizajner prvo napravi osnovni prijedlog koji se sastoji od koncepta, liste značajki, priče, publike na koju se cilja, osnovne zahtjeve, osoblje te potreban budžet. Može koristiti scripting language kako bi implementirao i pregledao ideje za dizajne bez mijenjanja koda igre. [4]

Prototip je nedovršena verzija programa ili igre koja se razvija. Završni proizvod može biti potpuno drukčiji od prototipa.

Proces uključuje:

1. Identificiranje osnovnih zahtjeva
2. Razvoj prvog prototipa
3. Recenzija
4. Izmjeniti prototip

Čest naziv za sučelje prototipa je horizontalni prototip. Omogućava pregled cijelog sistema fokusirajući se na interakciju korisnika. Horizontalni prototip je koristan za potvrdu korisničkog sučelja zahtjevima sustavima, demonstraciju verzije sustava za dobivanje *buy-in* poslovanja, razvijanje preliminarne procjene vremena razvoja i troškova.

Vertikalni prototip je detaljnija razrada jednog podsustava ili funkcije. Koristan je za dobivanje detaljnih zahtjeva za određene funkcije sa sjedećim koristima: uređivanje dizajna baze podataka, prikupljanje informacije o količinama podataka i potrebama sučelja te pojašnjavanje kompleksnih zahtjeva. [4]

2.3.2. Produkcija

U produkciji, programeri pišu novi izvorni kod, dizajneri razvijaju karakteristike igre kao 3D modele. Inžinjeri zvuka razvijaju zvučne efekte dok skladatelji razvijaju glazbu, dizajneri nivoa rade nivoe u igri, dok pisci pišu dijaloge koji će se odvijati. [4]

U ovoj fazi produkcije igre, definiraju se sadržaj, pravila, dizajn svijeta u kojem će se priča odvijati te sami likovi. [4]

Discipline:

- Dizajn svijeta: stvaranje priče, smještanje lika, tema igre.
- Dizajn sistema: stvaranje pravila
- Dizajn sadržaja: stvaranje likova, predmeta, misija
- Pisanje igre: pisanje priče, dijaloga
- Dizajniranje nivoa: stvaranje nivoa i njihovih svojstava
- Dizajn korisničkog sučelja: konstruiranje interakcija, povratnih podataka
- Dizajn zvuka: stvaranje glazbe i zvukova [4]

Programiranje je faza razvijanja programa video igre. Programeri često stvaraju prototip ideja i svojstava igre te su čestio dio dizajenrskog tipa koji dizajnira igru. Tijekom produkcije, programeri naprave veliki dio izvornog koda za stvaranje igre.

Izvorni kod se može generirati u skoro svakom tekst editoru, ali profesionalni programeri igara oriste *Integrated Development Environment,* npr. za Windows i Xpox se koriste *CodeWarrior* i *Microsoft Visual Studio.* [4]

Game art je proces stvaranja 2D i 3D modela za video igru. *Game artist* kreira umjetnički dio igre i model likova. [4]

Slika 18. Koncept modela za igru Half Life 2
[http://gamerant.com/halflife-2-episode-3-concept-art-dyce-157194/]

Slika 19. Koncept okruženja za igru Half Life 2
[http://gamerant.com/halflife-2-episode-3-concept-art-dyce-157194/]

Ilustracije igre, koje su uključene u raličite oblike medija (demo i *screenshot*), imaju značajan utjecaj na kupce pošto se ilustracije mogu ocjenjivati. Predmeti potrebni da se napravi dizajn i ilustracije variraju od olovke i gumice do potpunih programa za 2D i 3D dizajn. [4]

2.3.3. Post-produkcija

Igre za igraće konzole gotovo nikad nisu bile održavane nakon što su stupile na tržište, što je značilo da ako su imale ikakve greške, to nitko nije ispravljao. Danas, programeri održavaju i popravljaju sve greške u igrama tako da izdaju "zakrpe" koje korisnici mogu skinuti sa interneta te potom instalirati na svoje računalo i igra više ne bi imala grešaka. Programeri skupljaju dojave o greškama prije nego ih idu ispravljati. "Zakrpe" se mogu razvijati mjesecima te mogu sadržavati drukčije predmete ili misije. Kada se radi o *massively multiplayer online* igrama, održavanje počinje odmah sa isporukom. Održavanja se stalno radi pošto se *online* igre stalno mijenjaju i te se sa njima i mijenjaju predmeti i misije u igri. [4]

Izdavač igre cilja određeno tržište na koje će plasirati igru te je onda reklamira. Igra se reklamira tako da se u promotivni materijal stavi visoka grafika igre te se sve stavlja u časopise ili se promovira na TV-u. Također se natržište stavlja demo igre, odnosno verzija igre sa ograničenjima. Demo može trajati par sekundi ili korisnik igru može igrati par sati te je namijenjen novinarima koji pišu recenzije igara, sajmovima igara, općoj publici te internim zaposlenicima koji trebaju upoznati igru kako bi je promovirali. [4]

2.4. Programski alati

Programske alate koriste programeri za razvoj igre, ispravljanje grešaka i održavanje. Naziv se inače odnosi na jednostavne alate koji se mogu kombinirati zajedno.
Alati koji se često upotrebljavaju su:

- Sastavljači: programi koji dopuštaju pretvaranje izvornog koda u izvršnu datoteku
- Editori teksta: dopuštaju stvranje tekst datoteke koja sadržava izvorni kod. To može biti jednostavni "Notepad" ili neki složeniji program.
- Ispravljači grešaka: pronalaze probleme u izvornom kodu zbog kojih se program ponaša različito od onoga kako bi trebao (iznenadno gašenje). Ovakav program dopušta programeru ulazak u izvorni kako bi vidio točno koji dio koda ne radi kako treba. [5]

2.4.1. Objektno orijentirano programiranje

Objektno-orijentirano programiranje (OOP) je programska paradigma koja predstavlja pojam "objekata" koji imaju polja podataka (atributa koji opisuju objekt) i popratne postupke poznate kao metoda. Objekti, koji su obično slučajevi klase, se koristi za interakciju s drugima osmisliti programe i računalne programe.

U posljednjih nekoliko godina, objektno-orijentirano programiranje je postalo posebno popularano u dinamičnim programskim jezicima. Python, Ruby i Groovy su dinamički jezici izgrađeni na načelima OOP.

Simula (1967) je prihvaćen kao prvi jezik s osnovnim značajkama objektno-orijentiranog jezika. Stvoren je za izradu simulacijskih programa, u kojem su predmeti najvažnija informacija.

Neki od drugih OOP jezika su: Dylan, Eiffel, Falcon, F-Script, Gambas, Ruby, REALbasic. [5]

2.4.2. Programski jezici

Programski jezik je formalno konstruiran jezik dizajniran kako bi komunicirao instrukcije strojevima, odnosno računalima. Opis programskog jezika je podijeljen na dvije komponent sintakse i semantike, odnosno forme i značenja. Neki jezici su definirani određenim dokumentom, npr. C programski jezik je određen ISO standardima. [5]

Kada se rade igrice, nakon što se odredi dizajn, treba se odrediti kojim će se jezikom igra napraviti. Izbor ovisi o mnogo faktora, kao koji jezik poznaje programsko osoblje i na koje platforme će se igra staviti. Danas, pošto je objektno orijentiran, te binarno usklađen, najpopularniji programski jezik za igre je C++. Java i C jezici su također dobri, ali su neprikladni za neke stvari. [5]

C programski jezik je jezik opće namjene, te se u njemu može napraviti gotovo sve: rješavanje zadataka, pisanje *drivera*, operacijskih sustava, teksta procesora ili igara. Omogućuje i uključivanje naredbi pisanih asemblerski, zbog čega je zajedno s mogućnošću direktnog pristupa pojedinim bitovima, bajtovima ili cijelim blokovima memorije, pogodan za pisanje sistemskog softvera.

Struktura programa odnosi se na način pisanja programa i ako se ona ne poštuje, program neće raditi, bez obzira na to što su uporabljene sve potrebne naredbe. [6]

C++ je programski jezik različitih dijalekta, koji postoje zbog različitih komplajera. Dizajniran je sa sklonošću prema sistemskom programiranju, s performansama, učinkovitosti i fleksibilnosti korištenja i njegovim zahtjevima dizajna. Također je koristan u desktop aplikacijama, web pretraživanju, telefosnkim prekidačima te video igrama. C++ je sastavni jezik s implementacijama koje su dostupne na brojnim platformama. [6]

C# je izumljen s ciljem da .NET platforma dobije programski jezik koji bi maksimalno iskoristio njezine sposobnosti. Jezik treba osigurati podršku za program inženjerskih načela kao što su jake provjere. Robustnost softvera, trajsnot te programerska produktivnost su vrlo važni. Iako su C# aplikacije trebale biti ekonomične u pogledu memorije i snage procesora, jezik nije namjenjen da se izravno natječe performansama i veličinom sa C jezikom. [6]

Programi pisani u **Javi** se mogu izvoditi bez preinaka na svim operativnim sustavima za koje postoji *Java Virtual Machine (JVM)*, dok je klasične programe pisane npr. u C-u potrebno prilagođavati platformi. Iako je Java inspirirana C jezikom, ona pruža bolji stupanj sigurnosti ipouzdanosti zahvaljujući VM-u i hermetički zatvorenom okolišu u kome svaki program operira. Na Javi se programi brže razvijaju i to s manje pogrešaka. Ona je osnovni jezik za programiranje Googleovog sustava Android. [7]

2.4.3. API i knjižnice

U računalnom programiranju, aplikacijsko programsko sučelje (*application programming interface*), je skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama i resursima operacijskog sustava kao standardne biblioteke rutina, strukture podataka, objekata i protokola. Korištenjem API-ja omogućava programerima koristiti rad drugih programera štedeći vrijeme i trud koji je potreban da se napiše neki složeni program, pri čemu svi programeri koriste iste standarde. Napretkom u operacijskim sustavima, osobito napretkom u grafičkom korisničkom sučelju API je nezaobilazan u stvaranju novih aplikacija.

Ključna odluka u programiranju video igara je koje API i knjižnice koristiti. Neke knjižnice dobro podržavaju procesiranje zvuka ili renderiranje grafike. Odabir ovisi najviše o izboru platforme. Knjižnice za razvoj PlayStation 2, nisu dostupne za Microsoft Windows i obrnuto. No, postoje knjižnice koje olakšavaju razvoj igara koje se trebaju ponuditi na svim platformama, tako da programeri lako mogu napraviti igru u jednom jeziku te je mogu igrati na nekoliko platformi. [8]

Danas se najviše koristi 3D grafika, pa je tako najpopularniji API za 3D grafiku (za Microsoft Windows) Direct3D i OpenGL. DirectX je kolekcija igraćih API-ja. DirectX nije prijenosan, pa je napravljen OpenGL koji se može prenositi s platforme na platformu. Npr. igru Quake II, koja koristi OpenGL, je prenio sa Windowsa na Linux jedan obožavatelj igre. [8]

2.4.4. IDE Sučelja

Integrirano razvojno okruženje je program koji pruža opsežne mogućnosti računalnim programerima za razvoj softvera. IDE predstavljaju program koji pruža mogućnosti za pisanje, modificiranje, sastavljanje, implementaciju i ispravljanje pogrešaka. Jedan od ciljeva IDE sučelja je smanjiti konfiguraciju potrebnu za sastavljanje više razvojnih alata. [9]

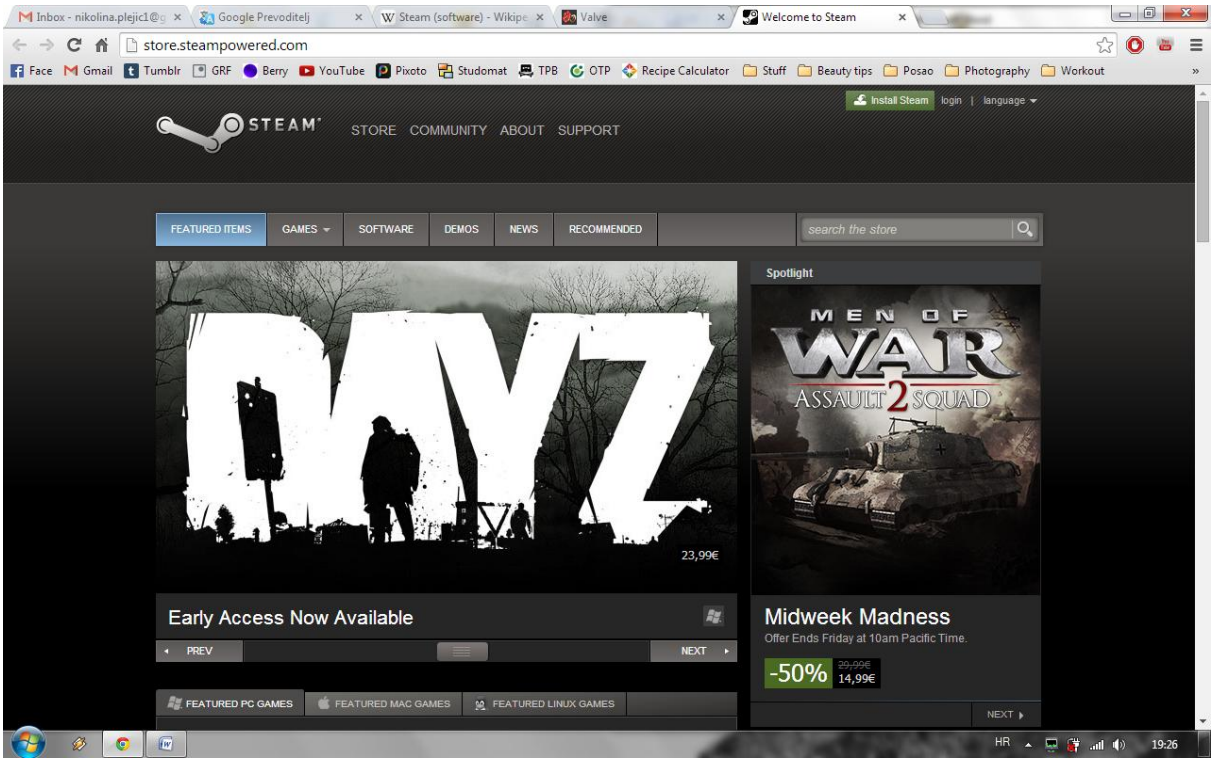
Visual Studio se koristi za razvoj aplikacija za *Windows Store*, desktop, mobilnih aplikacija, ASP:NET web aplikacija i XML web servisa. *Visual Studio* podržava različite programske jezike te dopušta uredniku koda da podržava bilo koji jezik. Ugrađeni jezici uključuju C, C++, C++/CLI, VB:NET i F#. Također podržava HTML/XHTML, XML/XSLT, JavaScript i CSS. [10]

2.5. Distribucija finalnog proizvoda - Marketing

Kako bi se olakšala prodaja igara, neke tvrtke su napravile svoje platforme (programe) za digitalnu distribuciju. Te platforme, kao Steam i Origin , pružaju usluge kupnje i preuzimanje digitalnog sadržaja za određenu igru. Neke platforme mogu služiti kao digitalni sustav upravljanja prava, ograničavajući tako kupljene stvari na jedan korisnički račun.

2.5.1. Steam

Steam je internetska platforma za digitalnu distribuciju i upravljanje digitalnim pravima, te društvena mreža napravljena od strane korporacije Valve. Steam korisniku pruža instalaciju igre te automatsko ažuriranje igara. Također podržava društvene značajke kao liste prijatelja, grupe, snimanje na Steam oblak (slično kao Google Drive), te međusobno komuniciranje korisnika unutar igre (Call Of Duty). Program pruža besplatno aplikacijsko programsko sučelje zvano *Steamworks*, koje programeri mogu koristiti za integraciju mnogih Steam-ovih funkcija, uključujući umrežavanje, dostignuća u igri, mikro-transakcije i podršku za korisničke sadržaje kroz Steam-ovu radionicu. [11]



Slika 20. Izgled Steam-ovog sučelja
[http://store.steampowered.com/]

Na Steamu, neovisni programeri mogu postavljati svoje igre i pustiti korisnicima da ih ocijene. Ako igra postigne određeni broj pozivitnih ocjena, igra se može prodatavi na Steam-u. Programeri tako postavljaju slike kako njihova igra izgleda na određenoj konzoli, informacije o igri ili čak i videe koji služe kao uvodi u igru i da bi se pokazalo

kako se igra može igrati. Sve igre na Steam-u su namjenjene računalima ili drugim konzolama kao PlayStation.

Nakon što se igra stavi na Steam Greenlight, potebno je izraditi marketing plan te isti provesti, kako na Steam-u tako i na osobnim stranicama i društvenim mrežama kako bi se privuklo ljude da glasaju za odobravanje igre od strane Steam-a. [11]

2.5.2. Google Play

Google Play, ili Trgovina Play, je platforma za digitalnu distribuciju mobilnih igara za mobitele koji imaju opercaijski sustav Android. Osim što nudi igre, također ima veliku trgovinu časopisa, glazbe, filmova, televizijskih programa i knjiga.Na Trgovini se mogu kupiti i uređaju kao Chromebooks i Google Nexus mobilni uređaji.

Pošto se i na Trgovini Play mogu dodati vlastitie igre, Android nudi nekoliko marketinških trikova za programere kako bi njihova igra postigla veću prodaju. [12]

2.5.3. I-Tunes

I-Tunes, kao i Trgovina Play, nudi mogućnost skidanja glazbe, filmova, glazbenih spotova, tv serija i audio knjiga, ali je dostupna samo na Iphone i iPod Touch proizvodima. I-Tunes se može skinuti na Mac i Windwos računala, te funkcionira kao program na kojem svira glazba gdje postoji i trgovina na kojoj se može kupiti glazba. [11]

2.5.4.Social networks

Kako bi aplikacija ili igra uspjela, puno ljudi radi poslovne stranice na Facebook-u. Stranica se podijeli sa prijateljima koji djele sa svojim prijateljima i tako dalje. Poslovna stranica je na Facebook-u besplatna, no samo sa time, stranica neće imati puno pogleda. Zato postoje oglasi, koje kada korisnik uđe u svoj račun može vidjeti sa desne strane. Oglasi se naFacebook-u plaćaju od 5$ do 20$ po danu ovisno o buđetu.

Osim Facebook-a, vrlo je efikasno oglašavati igre i aplikacije na Twitter-u, čije oglašavanje radi na sličnom principu kao i Facebook. Oglašavati se može na Google+-u, te na LinkedIn-u. Jedna od drukčijih stranica za oglašavanje, StumbleUpon. Ova

stranica nema klasične oglase, nego se da link stranice na kojoj je igra, te se odredi broj posjetitelja koji trebaju vidjeti stranicu. Svaki posjetitelj će koštati 10 centa. Korisnici preko StumbleUpon dolazi na druge stranice nasumično, ali prema kategorijama koje su odabrali. [13]

3. PRAKTIČNI DIO

3.1. Razvoj igre

Zbog jednostavnosti i kompaktabilnosti sa većinom današnjih uređaja koji imaju ekran na dodir, odnosno mobitela i tableta, razvoj ove igre vrši se u Android operativnom sistemu za mobilne uređaje. Android je danas, uz iOS (mobilni operativni sistem proizveden od strane Apple inc. kompanije) najrasprostranjeniji i najkorišteniji operativni sustav. Taj operativni sustav je osmišljen primarno za korištenje s ekranima na dodir. Svake godine, unapređuje se sustav, u cilju poboljšanja korisničkog iskustva pri radu s takvim ekranima.

Kao alat za izradu, koristi se Android Studio, program za razvoj aplikacija baziran na IntelliJ IDEA sustavu. Taj sustav se koristio nekoliko godina, te je na kraju otkupljen od strane Google-a, te iskorišten kao podloga za razvoj novog sučelja koji postaje novi standard u tom polju.

Kao tema igre, izabrana je „Whack – a – Mole“ ideja. Koncept igre je baziran upravo na starim arkadnim igrama iz ere prvih arkadnih igara, te je do danas ta igra izvedena u bezbroj različitih verzija. Igra se sastoji od pozadine sa 8 polja. Iz tih polja izlaze objekti koje je potrebno „udariti“ prije nego nestanu sa pozadine. Igra je bazirana na vremenskom principu, odnosno, što više se više objekata „udari“ u određenom vremenu, to je bolji rezultat.

3.2. Programski kod igre

Kod je napisan u Java programskom jeziku. Java se danas koristi kao jedan od najrasprostranjenijih jezika, baš zbog toga što je veoma prikladan za početnike, da bi krenuli raditi u polju programiranja. Postoji mnoštvo literature, te prigodnih tutorijala preko kojih se može u kratkom vremenu svladati osnove programiranja.

Kod je podijeljen u 5 klasa, „Main Activity“, „Score“, „Time“, „Utility“, „WhackView“ (imena klasa su dodjeljena proizvoljno, korišten je engleski jezik, da pojednostavimo kodiranje, odnosno da bi bilo jednostavnije komunicirati u Java programskom jeziku i da bi taj kod bio razumljiv u globalnom pogledu). Svaka od tih klasa je zasebni dio koda zajedno tvore cijelinu koja svojim izvršavanjem

ukomponirava sve klase, komunicira među klasama, te na kraju imamo finalni proizvod, u ovom slučaju, igru. Dijelovi teksta nakon „//“ oznaka u tom redu, su komentari, koji nisu dio koda, nego su u njima zapisana objašnjenja dijelova koda. Komentari u kodu se inače koriste kao sredstvo pomoću kojeg programer može objasniti dijelove koda.

- **MAIN ACTIVITY**

Glavna klasa iz koje se pokreće igra.

```
package net.home.whackgame;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends Activity {
   //varijabla koja definira vrijeme do početka igre
   private final int startTime = 2000;
   //varijabla koja definira vrijeme ažuriranja igre
   private final int updateTime = 100;

   //varijabla koja broji ažuriranja
   private int updateCount;

   //varijabla koja broji vrijeme
   private int timeCount;

   //varijabla koja označava kontrolu za prikaz vremena
   private TextView timeText;

   //varijabla koja označava kontrolu za prikaz bodova
   private TextView scoreText;
   //lista sa poljima igre
   private final List<WhackView> mWhackList = new ArrayList<WhackView>();

   //varijabla koja pokreće cijelu igru pomoću uputa
   private final Handler mHandler = new Handler();
```

```
//varijabla sa uputama za igru
private final Runnable mRunnable = new Runnable() {
   @Override
   public void run() {
      updateCount++;

      updateTimeText();

      updateScoreText();

      int size = mWhackList.size();
      int random = Utility.getRandomInRange(0, size);

      for(int i = 0; i < size; i++) {
         WhackView whackView = mWhackList.get(i);

         whackView.update();

         if(updateCount >= 5) {
            if (i == random) {
               if (!whackView.isActive()) {
                  whackView.activate();

                  updateCount = 0;
               }
            }
         }
      }

      if(!isFinished()) {
         mHandler.postDelayed(this, updateTime);
      }
      else {
         mHandler.removeCallbacks(this);

         disableWhackViews();

         finishLayout.setVisibility(View.VISIBLE);
      }
   }
};

//varijabla koja oznacava mali izbornik koji se pokaže na kraju
LinearLayout finishLayout;

//Android metoda koja se poziva prilikom kreiranja activity-a
@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        updateCount = 0;

        timeCount = 0;

        timeText = (TextView) findViewById(R.id.timeText);

        scoreText = (TextView) findViewById(R.id.scoreText);



        mWhackList.add((WhackView) findViewById(R.id.whack1));
        mWhackList.add((WhackView) findViewById(R.id.whack2));
        mWhackList.add((WhackView) findViewById(R.id.whack3));
        mWhackList.add((WhackView) findViewById(R.id.whack4));
        mWhackList.add((WhackView) findViewById(R.id.whack5));
        mWhackList.add((WhackView) findViewById(R.id.whack6));
        mWhackList.add((WhackView) findViewById(R.id.whack7));
        mWhackList.add((WhackView) findViewById(R.id.whack8));

        finishLayout = (LinearLayout) findViewById(R.id.finishLayout);

        Button btnPositive = (Button) findViewById(R.id.btnPositive);
        btnPositive.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finishLayout.setVisibility(View.GONE);

                enableWhackViews();
                reset();
                start();
            }
        });

        Button btnNegative = (Button) findViewById(R.id.btnNegative);
        btnNegative.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onBackPressed();
            }
        });
    }

    //Android metoda koja se poziva prilikom nastavka rada activity-a
    @Override
    protected void onResume() {
```

```
    super.onResume();

    if(isFinished()) {
        enableWhackViews();
        reset();
    }

    start();
}


 //Android metoda koja se poziva prilikom prestanka rada activity-a
@Override
protected void onPause() {
    super.onPause();

    mHandler.removeCallbacks(mRunnable);
}

//ažuriranje prikaza vremena
private void updateTimeText() {
    timeCount++;

    if(timeCount >= 10) {
        Time.subtract();

        timeText.setText(Integer.toString(Time.get()));

        timeCount = 0;
    }
}

//ažuriranje prikaza bodova
private void updateScoreText() {
    scoreText.setText(Integer.toString(Score.get()));
}

//provjera jeli igra gotova tj. jeli vrijeme isteklo
private boolean isFinished() {
    return Time.get() <= 0;
}

//start igre
private void start() {
    mHandler.postDelayed(mRunnable, startTime);
}

//resetiranje nekih parametara igre
```

```
    private void reset() {
        updateCount = 0;
        timeCount = 0;

        Score.reset();
        Time.reset();
    }

    //metoda za isključiti polja
    private void disableWhackViews() {
        for(WhackView whackView : mWhackList) {
            whackView.disable();
        }
    }

    //metoda za uključiti polja nakon isključivanja
    private void enableWhackViews() {
        for(WhackView whackView : mWhackList) {
            whackView.enable();
        }
    }
}
```

- **SCORE**

Jednostavna klasa koja nam služi za upravljanje s bodovima.

```
package net.home.whackgame;

public class Score {
    //varijabla sa bodovima
    private static int mScore = 0;

    //metoda za dohvaćanje bodova
    static int get() {
        return mScore;
    }

    //metoda za dodavanje bodova
    static void add() {
        mScore = mScore + 1;
    }

    //metoda za resetiranje bodova
    static void reset() {
        mScore = 0;
    }
```

```
}
```

- **TIME**

Jednostavna klasa koja nam služi za upravljanje proteklim vremenom.

```
package net.home.whackgame;

public class Time {
   //varijabla s preostalim vremenom u sekundama
   private static int mTime = 30;

   //metoda za dohvaćanje vremena
   static int get() {
      return mTime;
   }


   //metoda za oduzimanje vremena - sekundi
   static void subtract() {
      mTime = mTime - 1;
   }

   //metoda za resetiranje vremena
   static void reset() {
      mTime = 30;
   }
}
```

- **UTILITY**

Klasa Utility nam služi za dohvaćanje jednog nasumičnog broja. Metoda koju koristimo unutar ove klase je deklarirana kao "static" što znači da nije potrebno instancirati objekt klase Utility kako bi je upotrijebili.

```
package net.home.whackgame;

import java.util.Random;
public class Utility {
   //varijabla koja služi za stvaranje random brojeva
   private final static Random mRandom = new Random();

   //metoda za generiranje random broja u rasponu min - max
   public static int getRandomInRange(int min, int max) {
      return mRandom.nextInt((max - min) + 1) + min;
   }
```

```
}
```

- **WHACKVIEW**

Ova klasa je naslijeđena iz Android-ove klase ImageView, a služi za prikaz i upravljanje s iskakajućim elemenata na ekranu. Klasa ImageView je inače Android UI kontrola koja se koristi za prikaz slike na ekranu, a mi smo joj dodali logiku za iskakajuće elemente.

```
package net.home.whackgame;

import android.content.Context;
import android.util.AttributeSet;
import android.view.View;
import android.widget.ImageView;

public class WhackView extends ImageView {
   //varijabla koja označava stanje polja igre
   private int state;

   //varijabla koja broji ažuriranja polja
   private int updateCount;

   //konstruktor
   public WhackView(Context context) {
      super(context);
      initialize();
   }


    //konstruktor
   public WhackView(Context context, AttributeSet attrs) {
      super(context, attrs);
      initialize();
   }

   //konstruktor
   public WhackView(Context context, AttributeSet attrs, int defStyleAttr) {
      super(context, attrs, defStyleAttr);
      initialize();
   }

   //metoda za postavljanje početnih parametara polja
   private void initialize() {
      reset();
```

```
        setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if(state == 1) {
                    destroy();

                    Score.add();
                }
            }
        });

        setSoundEffectsEnabled(false);
    }

    //metoda koja se poziva prilikom svakog ažuriranja polja
    void update() {
        if(isActive()) {
            updateCount++;

            if (state == 1 && updateCount >= 15) {
                removeImage();

                if(updateCount >= 20) {
                    reset();
                }
            } else if (state == 2 && updateCount >= 5) {
                removeImage();

                if(updateCount >= 10) {
                    reset();
                }
            }
        }
    }
    //metoda za aktiviranje polja
    void activate() {
        state = 1;

        setImageResource(R.drawable.android_final_1);
    }



//metoda za uništenje polja, poziva se prilikom klika na polje
    private void destroy() {
        state = 2;
```

```
            updateCount = 0;

            setImageResource(R.drawable.android_final_2);
        }

        //metoda za isprazniti sliku u polju
        private void removeImage() {
            setImageDrawable(null);
        }

        //metoda za resetiranje polja
        private void reset() {
            state = 0;

            updateCount = 0;
        }

        //metoda koja provjerava jeli polje aktivno
        boolean isActive() {
            return state != 0;
        }

        //metoda za isključivanje polja
        void disable() {
            setEnabled(false);
        }

        //metoda za uključivanje polja
        void enable() {
            removeImage();

            reset();

            setEnabled(true);
        }
    }
```

- **UPUTE**

Ovdje je objašnjena petlja tzv. „Game Loop“, odnosno petlja koja „vrti“ igru. Ovaj dio koda se nalazi u Main Activity-u.

```
//varijabla sa uputama za igru
private final Runnable mRunnable = new Runnable() {
    @Override
    public void run() {
        updateCount++;

        //ažuriranje teksta kontrole s vremenom
        updateTimeText();

        //ažuriranje teksta kontrole s bodovima
        updateScoreText();

        //spremanje veličine liste polja
        int size = mWhackList.size();

        //generiranje random broja pozivom metode iz klase Utility i spremanje istog
          u varijablu
        int random = Utility.getRandomInRange(0, size);

        //petlja koja prođe kroz sve elemente u listi
        for(int i = 0; i < size; i++) {
        //spremanje dotičnog polja u varijablu
            WhackView whackView = mWhackList.get(i);

        //poziv metode update() od polja
        //update() se brine za stanje polja (da li je polje kliknuto, treba li maknuti
sličicu...)
            whackView.update();

        //provjera jeli prošlo više od 500ms od zadnjeg aktiviranja nekog polja
            if(updateCount >= 5) {
                //ako je, provjera da li se index trenutnog polja u
                 petlji podudara sa generiranim random brojem
              if (i == random) {
                //ako da, provjera jeli polje već aktivno, i ako nije, aktivacija istog
                if (!whackView.isActive()) {
                //aktivacija polja tj. postavljanje stanja polja na 1
                    whackView.activate();
                        //resetiranje brojača ažuriranja
                    updateCount = 0;
                }
              }
            }
```

```
        }

                        //provjera jeli igra gotova (igra je gotova kad je vrijeme isteklo)
                        //ako nije, nastavi sa izvršavanjem uputa
        if(!isFinished()) {
           mHandler.postDelayed(this, updateTime);
        } //ako je, prekini izvršavanje uputa, isključi polja i prikaži dijalog za
ponavljanje igre
        else {
           mHandler.removeCallbacks(this);

           disableWhackViews();

           finishLayout.setVisibility(View.VISIBLE);
        }
     }
  };
```

- **ACTIVITY MAIN.xml**

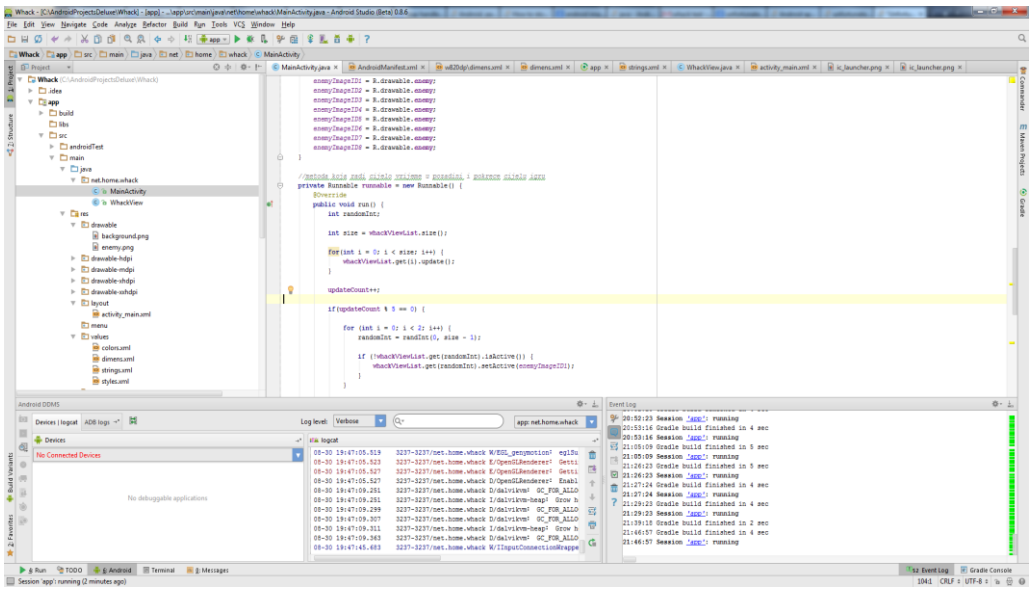
Ovdje je definiran raspored vizualnih elemenata igre. Inače, za android aplikacije, koristi se XML (Extensible Markup Language) za definiranje raspored elementa, odnosno „izgled“ aplikacije.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:background="@drawable/background"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <TextView
      android:id="@+id/timeText"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="start|top"
      android:layout_marginStart="20dp"
      android:layout_marginLeft="20dp"
      android:layout_marginTop="20dp"
      android:textSize="25sp"
      android:textColor="@android:color/white" />
```

```
<TextView
    android:id="@+id/scoreText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="top|center"
    android:layout_marginTop="20dp"
    android:textSize="50sp"
    android:textColor="@android:color/black" />

 <net.home.whackgame.WhackView
    android:id="@+id/whack1"
    android:layout_width="@dimen/whack_width"
    android:layout_height="@dimen/whack_height"
    android:layout_marginStart="25dp"
    android:layout_marginLeft="25dp"
    android:layout_marginTop="120dp" />
 <net.home.whackgame.WhackView
    android:id="@+id/whack2"
    android:layout_width="@dimen/whack_width"
    android:layout_height="@dimen/whack_height"
    android:layout_marginStart="172dp"
    android:layout_marginLeft="172dp"
    android:layout_marginTop="112dp" />
 <net.home.whackgame.WhackView
    android:id="@+id/whack3"
    android:layout_width="@dimen/whack_width"
    android:layout_height="@dimen/whack_height"
    android:layout_marginStart="324dp"
    android:layout_marginLeft="324dp"
    android:layout_marginTop="110dp" />
 <net.home.whackgame.WhackView
    android:id="@+id/whack4"
    android:layout_width="@dimen/whack_width"
    android:layout_height="@dimen/whack_height"
    android:layout_marginStart="472dp"
    android:layout_marginLeft="472dp"
    android:layout_marginTop="124dp" />
 <net.home.whackgame.WhackView
    android:id="@+id/whack5"
    android:layout_width="@dimen/whack_width"
    android:layout_height="@dimen/whack_height"
    android:layout_marginStart="25dp"
    android:layout_marginLeft="25dp"
    android:layout_marginTop="240dp" />
```

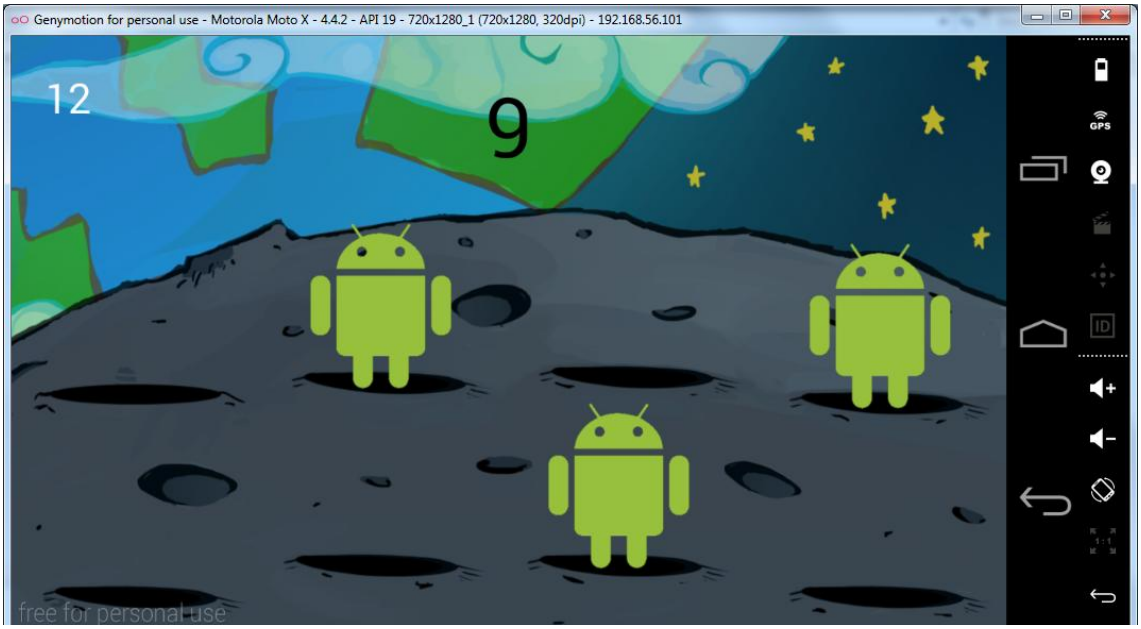
```
<net.home.whackgame.WhackView
    android:id="@+id/whack6"
    android:layout_width="@dimen/whack_width"
    android:layout_height="@dimen/whack_height"
    android:layout_marginStart="170dp"
    android:layout_marginLeft="170dp"
    android:layout_marginTop="220dp" />
 <net.home.whackgame.WhackView
    android:id="@+id/whack7"
    android:layout_width="@dimen/whack_width"
    android:layout_height="@dimen/whack_height"
    android:layout_marginStart="315dp"
    android:layout_marginLeft="315dp"
    android:layout_marginTop="220dp" />
 <net.home.whackgame.WhackView
    android:id="@+id/whack8"
    android:layout_width="@dimen/whack_width"
    android:layout_height="@dimen/whack_height"
    android:layout_marginStart="475dp"
    android:layout_marginLeft="475dp"
    android:layout_marginTop="240dp" />

 <LinearLayout
    android:id="@+id/finishLayout"
    android:background="@android:color/white"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:orientation="vertical"
    android:visibility="gone">

    <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="center"
       android:text="@string/play_again"
       android:textColor="@android:color/black"
       android:textSize="24sp" />

    <LinearLayout
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:orientation="horizontal">
       <Button
          android:id="@+id/btnPositive"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
```

```
            android:text="@string/play_positive" />

        <Button
            android:id="@+id/btnNegative"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/play_negative" />

    </LinearLayout>

</LinearLayout>

</FrameLayout>
```

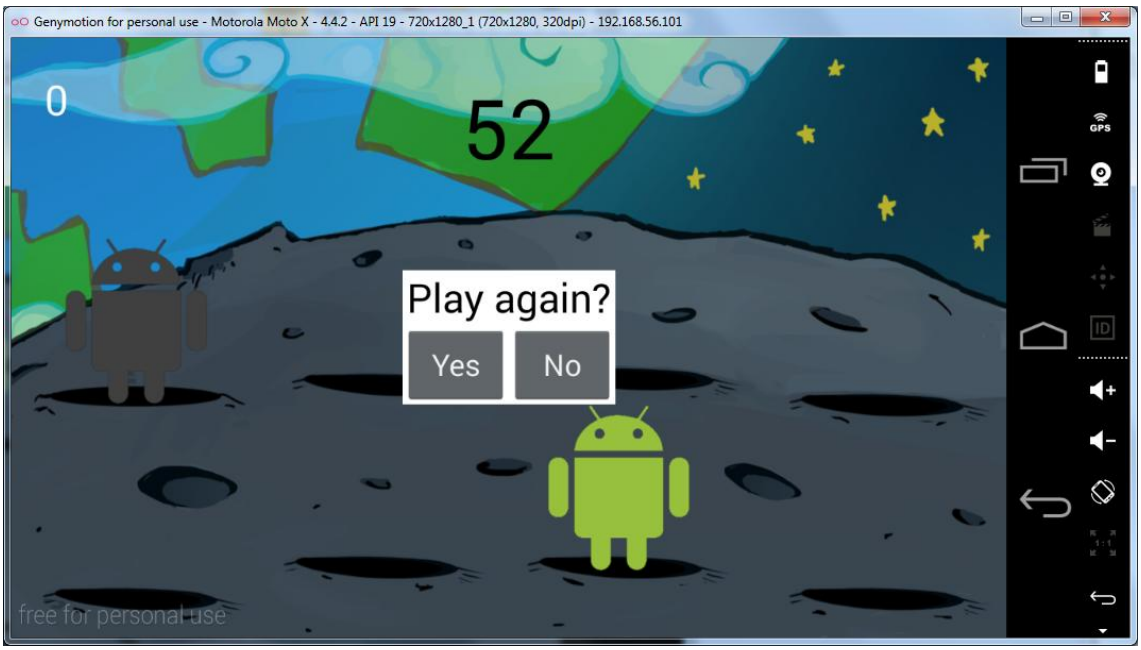


Slika 21. Radno okruženje Android Studija

Nakon što je kod napisan, vrši se funkcija „debug“ u android studiu. Ta funkcija omogućava da „Izvršimo“ kod, te ga pregledamo cijelokupnog, te program vrši pregled koda, govori nam ako je došlo do greške, te daje svoje komentare. Nakon toga program generira, u ovom slučaju .apk datoteku. Ta datoteka je standardizirana te prepoznata od svih android uređaja. Nakon toga dovoljno je tu datoteku staviti na računalo, tablet, mobitel, ili bilo koji drugi uređaj sa ekranom na dodir, koji ima Android operativni sustav, te ju pokrenuti. Tada uređaj preuzima daljnje izvođenje procesa, te kao finalni proizvod imamo, u ovom slučaju, igru.



Slika 22. Vizualni prikaz igre na emulatoru za Android operativne sustave (aktivni dio igre)



Slika 23. Vizualni prikaz igre na emulatoru za Android operativne sustave (kraj igre)

4. ZAKLJUČAK

Kako napreduje tehnologija, tako naravno dolazi do promjena u izvedbi procesa vezanih za struku programiranja, ali u osnovi, baza je uvijek ista. Programski jezici ne nastaju svaku godinu, već programi koji se danas koriste koristili su se i prije 10 godina. Zbog činjenice da je razvoj novog programskog jezika izrazito zahtjevan i skup posao, a postojeći programski jezici nude pregršt opcija koje još uvijek nisu iskorištene do svog maksimalnog potencijala, treba pokušati unaprijediti igre na druge načine, a ne samo u kodnom dijelu. Novi uređaji s ekranima na dodir su sve brži, sve je bolje rješena izvedba korištenja. To dovodi do toga da se kod ljudi javlja želja za inovacijama. U polju ekrana na dodir, moglo bi se reći, sve je već izmišljeno. Mogućnost prepoznavanja višestrukih dodira, gesti je već osnovna funkcija današnjih uređaja. Stoga proizvođači igara za takve uređaje moraju pronalaziti nove i uzbudljive načine igranja na takvim uređajima. Možemo primjetiti porast igara za mobilne uređaje u zadnjih nekoliko godina, te zbog te činjenice, i činjenice da svi novi mobilni uređaju imaju ekran na dodir, možemo reći da je potražnja za igrama na takvim uređajima dominantna naspram klasičnih puteva igranja poput starih arkadnih uređaja, ili pak modernih konzola. To naravno govori, da budućnost igara za ekrane na dodir nije upitna, te da ulaganje u razvoj istih je isplativ i veoma profitabilan način zarade.

5. LITERATURA

1. ***http://documentarystorm.com/a-brief-history-of-video-games/, Documentary storm, A Brief History of Video Games, 07.09.2015
2. ***http://www.hongkiat.com/blog/evolution-of-home-video-game-consoles-1967-2011/, Hongkiat, Evolution og home video game consoles: 1967-2011, 07.09.2015
3. ***http://www.tru-vumonitors.com/images/Touch_Screen_Basics.Comparisons.pdf, Tru-Vu Monitors, Touch screen basics, 07.09.2015
4. Bethke, Erik (2003), Game development and production, Wordware Publishing, Plano, Texas
5. ***http://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep#Composition, Code project, Introduction to object oriented programing concepts and more, 07.09.2015
6. ***http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf, Ecma Interantional, C++ Language Specifications, 07.09.2015
7. Herbert Schildt (2012), Java, a begineer's guide, 5th Edition, The McGraw Hill Companies, Inc., SAD
8. ***http://readwrite.com/2013/09/19/api-defined, ReadWrite, What APIs are and why they're important, 07.09.2015
9. ***https://eclipse.org/ide/, Eclipse, IDE, 07.09.2015
10. ***http://msdn.microsoft.com/en-us/library/ms165079.aspx, Microsoft, Library, 15.08.2014
11. ***http://techsavvyglobal.com/top-10-digital-distribution-platforms/, Tech Savy global, Top 10 digital distribution platforms, 07.09.2015
12. ***https://play.google.com/intl/en_us/about/index.html, Google play, About, 07.09.2015
13. ***http://www.pcworld.com/article/2030779/how-to-advertise-on-social-media-step-by-step-on-4-networks.html?page=2, PC world, How to advertise on social media step by step on 4 networks, 07.09.2015