UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Vidmar

Naravni uporabniški vmesnik za aplikacijo digitalne galerije

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Borut Batagelj

Ljubljana 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00349/2012
Datum: 05.09.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANDREJ VIDMAR**

Naslov: **NARAVNI UPORABNIŠKI VMESNIK ZA APLIKACIJO DIGITALNE GALERIJE**

NATURAL USER INTERFACE FOR DIGITAL GALLERY APPLICATION

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Interakcija z računalniki in drugimi elektronskimi napravami se seli iz klasičnih uporabniških vmesnikov na naravne uporabniške vmesnike, kjer uporabnik namesto klasičnih pripomočkov kot so tipkovnica in miška uporablja dotik ali kretnje. Takšni vmesniki so še posebej uporabni v okoljih kjer bi bili dodatni elementi moteči oziroma kjer želimo, da uporabnik komunicira z napravo na čim bolj naraven način.

Diplomant naj v svojem delu pregleda možnosti uporabe naravnih uporabniških vmesnikov v okolju galerije za primer prikazovanja umetniških del. V ta namen naj izdela aplikacijo za prikaz umetniških del, katero lahko upravljamo na naraven način s pomočjo kretenj.

Mentor:

viš. pred. dr. Borut Batagelj

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

ZAHVALA

Zahvaljujem se mentorju dr. Borutu Batagelju za strokovno pomoč in nasvete med pisanjem diplomske naloge.

Zahvaljujem se vsem, ki so mi bili ob strani in me podpirali med časom študija. Še posebej se zahvaljujem staršem za finančno pomoč in sošolcem s katerimi smo se lepo razumeli in si bratsko pomagali med študijem.

KAZALO

KAZALO SLIK

POVZETEK

Diplomska naloga se osredotoča na izdelavo in opis interaktivne aplikacije, katere namen je predstavitev umetniških del, kot so slike ali kipi in dodatnih informacij o njih obiskovalcem galerij. Aplikacija temelji na kinetičnem uporabniškem vmesniku, kjer se uporabnik z gibi rok prebija skozi seznam umetnin. To omogoča Microsoftov igralni pripomoček Kinect, katerega prvotna funkcija je nadzor iger z gibi in kretnjami na igralni konzoli Xbox 360. Za potrebe te inštalacije bi Kinect priklopili na računalnik z okoljem Windows 7, na katerem bi bili nameščeni gonilniki za upravljanje s Kinectom in sama aplikacija. Za prikaz slike bi uporabili LCD TV ali projektor.

Podobne prezentacije so bile že uporabljene na različnih kulturnih in družbenih dogodkih in so se izkazale za zelo priljubljene in zanimive za obiskovalce.

Ključne besede: Senzor Kinect, Microsoft Kinect SDK, umetnost, novi mediji, galerija, JavaScript, HTML, CSS, Microsoft Xbox 360, Nintendo Wii, Play Station, Sony, igre, uporabniški vmesniki, kinetični uporabniški vmesniki, naravni uporabniški vmesniki, manipulacija, IR kamera, VGA kamera, Libfreenect, OpenNI, CL NUI.

ABSTRACT

This diploma thesis is focused on the creation and description of an interactive application, which goal is to show artworks like paintings or sculptures and additional information about them to visitors of art exhibitions. The aplication is based on a kinetic user interface, where the user is browsing thru a list of artworks. This is possible by using Microsoft's game controller Kinect, which was at first used to play games on the Xbox 360 game console using only the movement of the human body. For the needs of our theasis we will use Kinect on a Windows 7 based PC where our aplication will be installed.

Similar presentations were used on different cultural and social events and were very interesting for the visitors.

Key words: Sensor Kinect, Microsoft Kinect SDK, art, new media, gallery, JavaScript, HTML, CSS, Microsoft Xbox 360, Nintendo Wii, Play Station, Sony, games, user interface, kinetic user interface, natural user interface, manipulation, IR camera, VGA camera, Libfreenect, OpenNI, CL NUI.

Slovar kratic:

Microsoft Kinect: Igralni pripomoček za igralno konzolo Xbox 360, ki omogoča igranje iger z uporabo gibov in govornih ukazov

Microsoft Xbox 360: igralna konzola, ki omogoča igranje iger, brskanje po spletu, klepet z prijatelji...

IR: infra red – infra rdeča svetloba

Spletna stran: dokument z nadbesedilom, ki ga prikaže brskalnik

Novi mediji: novodobni elektronski mediji

Uporabniški vmesnik: je komponenta preko katere ljudje (uporabniki) upravljajo z računalnikom

NUI: natural user interface – naravni uporabniški vmesnik

KUI: kinetic user interface – kinetični uporabniški vmesnik

VGA: video graphics array

3D: tridimenzionalni

SDK: softwere develpomnet kit – skupek orodij za razvoj aplikacij

HTML: Hyper Text Markup Lenguage – jezik za označevanje nadbesedil

CSS: Cascading Style Sheets - kaskadne stilske podloge

JavaScript: skriptni jezik namenjen spletnemu programiranju

1 UVOD

S hitrim razvojem računalnikov in elektronike se odpira veliko novih možnosti, kako izvesti določeno opravilo ali nalogo. To še posebej velja za uporabniške vmesnike, ki so doživeli velike spremembe: od prvotnega vnašanja ukazov z uporabo stikal do sodobnih zaslonov na dotik. Za cilj te diplomske naloge smo razvili interaktivno aplikacijo za potrebe prikaza umetniških del obiskovalcem razstav. Aplikacijo se nadzira samo z uporabo gibov rok, brez potrebe po uporabi standardnih vhodno/izhodnih naprav kot so miška ali zaslon na dotik. To omogoča uporaba Microsoftove naprave Kinect (slika 1-1), ki je prvotno služila za igranje iger z gibi telesa na igralni konzoli Xbox 360 in je bila kasneje predelana s strani računalniških navdušencev, da je lahko omogočila priklop na osebni računalnik in osnovno upravljanje z njim.

Aplikacija bo v osnovi omogočila prikaz seznama umetniških del z njihovim predogledom, tako da uporabnik vidi kaj izbira. Z izborom umetnine se pojavi njena povečana verzija. Ob tem pa je še omogočen prikaz dodatnih informacij o umetnini, kot je avtor, letnica nastanka, opis in komentarji.

SLIKA 1-1: Microsoft Kinect

1.1 UMETNOST

Že od začetka obdobja človeštva je človek čutil potrebo po izražanju svojega pogleda na svet, izkušenj, čustev... To je storil preko umetniških del. V zgodnjih časih so bile to preproste slike na kamnu iz pigmentov (slika 1-2), preprosta glasbila ali kipci... Skozi stoletja je umetnost napredovala in postajala bolj in bolj kompleksna z izumi novih tehnologij in eksperimentiranjem v nove smeri. Prave definicije kaj je umetnost ni, oziroma se spreminja od posameznika do posameznika. Definiciji si še najbolj približa Enciklopedija Britannica, ki takole definira umetnost: uporaba spretnosti in domišljije za ustvarjanje estetskih objektov, okolij ali doživetij, ki se lahko delijo z drugimi [1].

SLIKA 1-2: Jamska slika

Kolikor je različnih vej umetnsti je še več načinov kako umetniška dela prikazati. Stoletja nazaj je bilo navadnemu človeku mogoče umetnine si ogledati le v muzejih in razstavah, predstave je bilo možno gledati v gledališčih in glasbo poslušati na koncertih... Z razvojem novih tehnologij pa se je umetnost močno vključila v naša vsakdanja življenja. Največji mejnik je bil izum fonografa – predhodnika gramofona, ki je omogočal snemanje in predvajanje zvoka. Izumila ge je Tomas Edison leta 1878 in je omogočal poslušanje glasbe kar v domačih hišah. Velikega prelomnica je tudi izum radija, ki je omogočal prenos zvoka preko radijskih valov. Sprva je bil radio mišljen le za komunikacijo, vendar se je kasneje pojavila zamisel o uporabi radia kot medija. Prvi radijski prenos se je začel leta 1906, ko je radijski oddajnik začel predvajati glasbo (slika 1-3).

SLIKA 1-3: Spremljanje zgodnjih radijskih prenosov

Leta 1938 je radio povzročil paniko med poslušalci, ko je predvajal H. G. Wellsovo dramo Vojna svetov. Do takrat je veljalo, da radio sporoča samo novice in predvaja glasbo in posledično je tisoče poslušalcev zajela panika, saj so mislili da poteka invazija nezemljanov. Takrat je radio postal alternativa kinematografom in gledališkim predstavam.

1.2 INTERNET

Podobno kot radio, je bil tudi internet z začetka zamišljen kot komunikacijsko sredstvo. Njegov namen je bil ohraniti komukacijsko omrežje ameriške vojske v slučaju Sovjetskega napada z jederskim orožjem in povezavi super računalnikov različnih inštitutov za delitev podatkov o raziskavah in simulacijah. Kasneje je postal vse bolj razširjen med akadameskimi krogi in proti koncu hladne vojne tudi med domačimi uporabniki. Z vsakim dnem se je pojavljalo več spletnih strani. Z začetka so bile to le preproste strani (slika 1-4), ki so vsebovale predstavitve podjetij, prodajne kataloge, novice, vreme... Kasneje pa so se začele pojavljati strani kjer so ljudje objavljali svoje fotografije, slike, skulpture, video posnetke... Internet je tako postal del novih medijev.

SLIKA 1-4: Prva spletna stran verige restavracij s hitro hrano

1.2.1 NOVI MEDIJI

Novi mediji je splošno ime za vse novodobne elektronske medije (slika 1-5), ki se ločujejo od starih medijev, kot so časopis, radio, televizija... Slednji veljajo za statične, saj ni mogoča interakcija med uporabniki in medijem. Pri novih medijih pa je mogoča komunikacija z uporabo računalniških tehnologij. Razvoj novih medijev je tesno povezan s pojavitvijo osebnih računalnikov in interneta v vsakdnaje življenje v 80. letih prejšnjega stoletja.

Zgledi novih medijev so:

-spletne strani in skupnosti (portali, forumi...),

-elektronska pošta,

-digitalna fotografija,

-računalniške igre,

-klepetalnice, video konference, internetna telefonija,

-internetna televizija (IP TV).

Pretirana raba novih medijev lahko tudi vodi v čedalj večjo izoliranost uporabnikov in nastanku novodobnih odvisnosti.

SLIKA 1-5: Razpon novih medijev

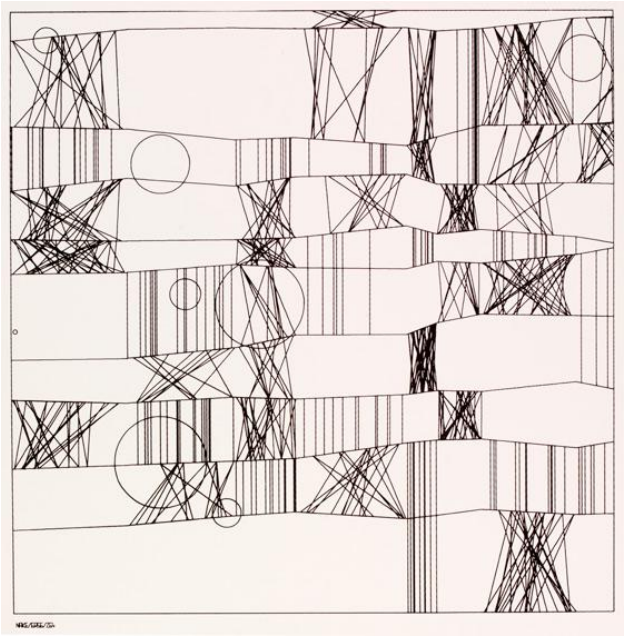
1.2.1.1 NOVI MEDIJI IN GLOBALIZACIJA

Zaradi možnosti interakcije imajo novi mediji velik vpliv na razvoj in širjenje svobodne demokracije in premagovanje kulturnih ovir in geografske odaljenosti. Dober primer tega je Arabska pomlad, ki se je začela decembra 2010. Njen temelj je bila ravno uporaba socialnih omrežij za organiziranje protestov in obveščanja tujih medijev o dogajanju v državah. Po drugi strani pa je lahko internet zlorabljen za širjenje sovražnega govora ter ekstremističnih in terorističnih mišljenj in dejavnosti.

Poleg širjenja govora pa novi mediji omogočajo tudi širjenje umetniških del in stvaritev. Včasih so morale biti novonastale glasbene skupine sprejete pod okrilje založniških hiš, da so lahko ponudile svojo glasbo poslušalcem. Dandanes pa to ni več potrebno, saj lahko skupine dosežejo publiko z vsega sveta z nalaganjem glasbe na portale kot so MySpace [2] ali YouTube [3]. Podobno velja za fotografe, slikarje, kiparje... ki lahko svoja dela objavljajo na ogled preko raznih spletnih poratlov kot so deviantART [4] ali depthCORE [5].

1.2.1.2 UMETNOST IN RAČUNALNIŠKE TEHNOLOGIJE

Skozi različne računalniške tehnologije se je mogoče tudi umetniško izražati. Začetki segajo v zgodnja 60. leta, ko so nastali prvi digitalni računalniki, predhodniki današnjih osebnih računalnikov. Do njih so lahko dostopali le pripadniki akadmenskih krogov, procesna moč je pa tudi bila zelo omejena. Ker ni bilo veliko aplikacij, so morali programerji pisati lastne, kar jim je omogočalo svobodo in raziskovanje v nove smeri. S pojavitvijo prvih primitivnih tiskalnikov se je odprla možnost širjenja informacije v papirnati obliki, v obliki teksta in slike. Slednje je bilo zelo težko realizirati, saj jih je bilo potrebno algoritmično definirati. Leta 1965 je Frieder Nake na tak način ustvaril sliko z imenom Hommage à Paul Klee 13/9/65 Nr.2 (slika 1-6), katera je veljala za najbolj kompleksno delo tistega časa [6].



SLIKA 1-6: Hommage à Paul Klee 13/9/65 Nr.2

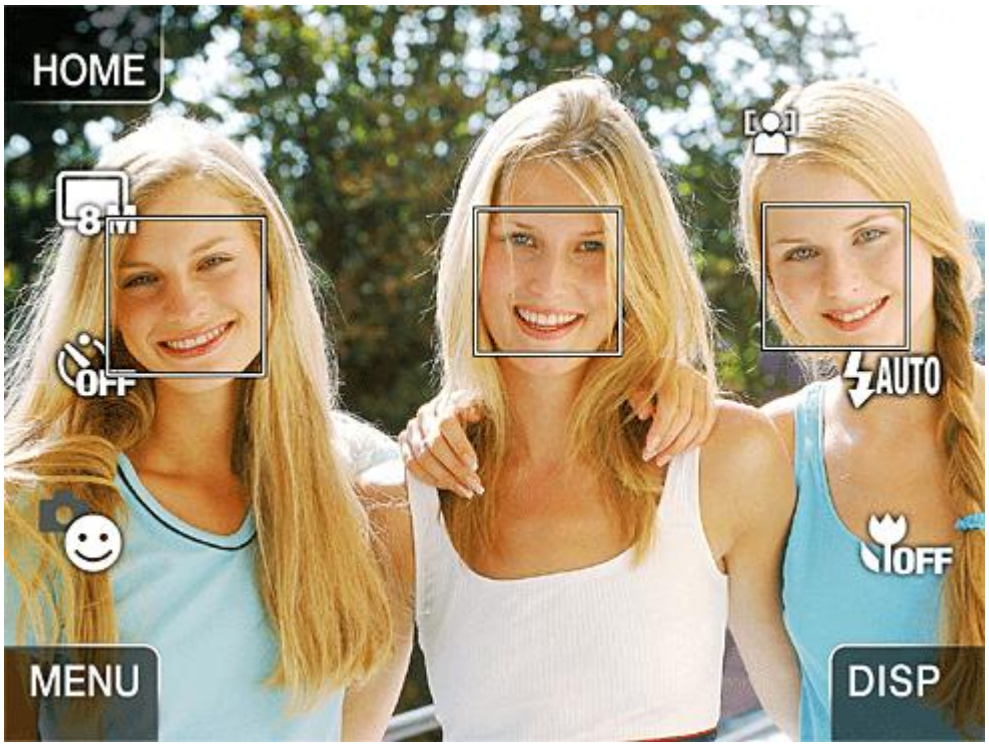
Z razvojem novih tehnologij skozi nadalna leta je prišlo do novih zamisli kako uporabiti računalnike v umetnosti. Pri Bell Laboratories so tako ustvarili prvo računalniško animacijo z tiskanjem podob na 35 milimeterski filemski trak.

V 80. letih so računalniške tehnologije prodrle v filmsko indrustrijo. Filmi kot so Star Trek II (1982) in Tron (1982) so uporabljali računalniške posebne efekte in grafiko (slika 1-7). Osebni računalniki so se množično začeli pojavljati po domovih, kar je povečalo povpraševanje po programski opremi. Med drugim so nastali programi za obdelavo slik, kot sta Adobe Photoshop in Ilustrator. Kasneje so se pojavili še programi za domačo rabo pri obdelavi video in avdio vsebin ter programi za ustvarjanje glasbe in 3D animacij. Dandanes naletimo na digitalno umetnost na skoraj vsakem koraku.

SLIKA 1-7: Prizor iz filma Tron (1982)

1.3 RAČUNALNIŠKI VID

Računalniški vid je veda ki se ukvarja s predstavitvijo slik ali video posnetkov realnega sveta v računalniku razumljivo obliko za nadaljno procesirjanje in uporabo. Cilj je posnemati človeško možnost dojemanja sveta preko oči, kar pa zahteva ogromno procesorsko moč za realnočasovno izvajanje. Do nedolgo nazaj je bilo to področje omejeno le na tiste korporacije, ki so bila pripravljena vlagati ogromno kapitala v zmogljive računalniške sisteme. V današjih časih pa doživljamo padec cen in naraščanje zmogljivosti mikroprocesorjev, kar omogoča množični prodor tehnologij, ki temeljijo na računalniškemu vidu v naše vsakodnevno življenje. Dober primer tega je možnost da že vsak digitalni fotoaparat omogoča detekcijo nasmeha na fotografirani osebi (slika 1-8). Računalniški vid se je množično začel pojavljati tudi v hitro rastoči panogi računalniških iger. Prvo funkcionalno napravo, ki je omogočala, da igralec upravlja z igro preko gibov rok s pomočjo računalniškega vida je razvil Logitech za igralno konzolo Play Station 2. To je bilo leta 2003. Naprava se je imenovala EyeToy in je bila v osnovi le spletna kamera s katero so posebej napisane igre prepoznale gibe igralca kot ukaze. Omenit še velja prihajajoči PlayStationEye in Microsoft Kinect, ki je bil izdan leta 2010 za konzolo Xbox 360. Slednji bo tudi osnova naše aplikacije.



SLIKA 1-8: Zaznavanje nasmeha na fotografiji

1.4 UPORABNIŠKI VMESNIKI

Uporabniški vmesnik je komponenta preko katere ljudje (uporabniki) upravljajo z računalnikom. Sestavljen je iz strojne in programske opreme. Uporabnik vnaša zahteve in podatke v računalnik preko vhodnih naprav, rezultate svojih dejanj in zahtev pa dobi preko izhodnih naprav. Poleg računalniških ved, so pri realiziaciji uporabniških umesnikov prisotne še druge vede: psihologija, ergonomija, grafično načrtovanje in inžinirstvo. Dober uporabniški vmesnik ne omogoči le surove funkcionalnosti, temveč je enostaven, efektiven in prijeten za uporabo. V današnjh časih je razvitih in uporabljenih že precej različnih vrst uporabniških vmesnikov. V tej diplomski nalogi se osredotočamo na precej novo podzvrst naravnih uporabniških vmesnikov (NUI – Natural User Interface): na kinetične uporabniške vmesnike (KUI – Kinetic User Interface), ki omogočajo nadzor nad napravimi s pomočjo gibov.

1.4.1 ZGODOVINA UPORABNIŠKIH VMESNIKOV

- obdobje med 1945 in 1968: Obdobje prvih računalnikov sodobnega tipa z možnostjo programiranja. Pojavljajo se batch uporabniški vmesniki – uporabnik je preko vhodnih naprav računalniku podal opravilo in parametre opravila. Po zagonu opravila je računalnik izvajal ukaze in operacije, dokler ni izračunal rešitve. Do takrat je bil neodziven za vnos in izvajanje drugih ukazov.
- obdobje po 1969: Razvoj zmogljivih centralnih procesnih enot in padec cen pomnilnika je pripeljal do možnosti opravljanja več opravil (aplikacij) hkrati . Pojavi se ukazna vrstica – tekstovno okno, kjer uporabnik preko tipkovnice vpisuje ukaze, katere računalnik izvede. Rezultat je predstavljen kot tekst na zaslonu. Skoraj vsi moderni operacijski sistemi ohranjajo ukazno vrstico.
- odbdobje od 1981: Čeprav je načrtovanje konceptov grafičnega umesnika potekalo že od 40ih let, so komaj v zgodnjih 80ih letih prišle na tržišče prve uspešne komercialne izvedbe. Grafični vmesniki omogočajo interakcijo z napravo preko grafičnih gradnjikov, kot so ikone, okna in meniji. V ta namen se uporabljajo različne vhodne naprave, kot so miška, tipkovnica, zasloni na dotik,...

Vredno je omeniti leta 1973 izdelani računalnik Xerox PARC – Alto: prvi računalnik, ki je uporabljal koncept namizja in grafični uporabniški vmesnik (slika 1-9). Kot prvi je tudi omogočal interakcijo z uporabo računalniške miške. Danes velja za temelj vseh sodobnih uporabniških vmesnikov, saj vsebuje skoraj vse gradnike, ki so še danes v uporabi. Sistem ni bil komercialno usmerjen, je bil pa uporabljen v Xereroxovih pisarnah in po univerzah. Naziv prvega komercialno uspešnega računalnika z grafičnim vmesnikom je prevzel Apple.

SLIKA 1-9: Uporabniški vmesnik na sistemu Xerox PARC – Alto

1.4.2 NARAVNI UPORABNIŠKI VMESNIKI

Naravni uporabniški vmesniki (NUI) so poiskus uporabe uporabniku že poznanih načinov interakcije z realnim svetom za upravljanje z računalniškimi tehnologijami. Uporabnik tako nima potrebe se naučiti upravljati sistemov z uporabo klasičnih vhodnih naprav kot so tipkovnica ali miška, ampak zgolj prenaša že osvojena znanja in intuicijo na novo področje. To lahko počne preko mnogo novih tehnologij kot so zasloni na dotik, 3D kamere, sistemov za prepoznavanje govora, sistemov s senzorji gibanja in pospeškov... Trenutni uporabniški vmesniki temeljijo na konceptu WIMP – windows, icons, menus, pointers, ki sega v leto 1973 in so za upravljanje mnogih današnjih naprav kot so pametni telefoni, industriski računalniki ali vojaški računalniški sistemi precej nerodni in počasni. NUI pa te omejitve odpravijo. Dober zgled je operacijski sistem Android, ki ga najdemo na pametnih telefonih in tabličnih računalnikih (slika 1-10).

SLIKA 1-10: Uporabniški vmesnik na Androidu

1.4.3 ZGODOVINA KINETIČNIH UPORABNIŠKIH VMESNIKOV

Ideja kinetičnih uporabniških vmesnikov je v resnici precej stara. V kombinaciji z navidezno resničnostjo (angl. Virtual Reality), ki jo lahko zasledimo že v romanih o znanstveni fantastiki v obdobju pred drugo svetovno vojno. Prvi funkcionalni prototip kinetičnih vmesnikov oziroma naprave pa so se pojavili leta 1977. Rokavico, ki je služila kot vhodna naprava za delo z računalnikom so razvili v laboratoriju za elektronsko vizualizacijo (Electronic Visualization Laboratory) na Univerzi v Chicagu, Združene Države.

Rokavica je bila opremljena s senzorji, ki so spremljali krčenje in raztezanje sklepov na prstih človeške roke. Na podlagi parametrov je programska oprema prevedla različne položaje prstov v ukaze, ki jih je računalnik izvedel. Zaradi visoke cene in pomankanja tržišča ni bilo izdelanih veliko primerov digitalnih rokavic.

Prvo komercialno rokavico, ki je omogočala realnočasovno delo z računalniškim sistemom je bila izdelana šele leta 1987 pri družbi Abrams/Gentle Entertainment kot igralni pripomoček za igralno konzolo Ninetndo Entertainment System. Imenovala se je Nintendo Power Glove (slika 1-11). Poleg spremljanja gest s prsti, je rokavica omogočala še vnos podatkov z uporabo gumbov na njej in možnost zaznavanja pozicije roke v razmerju s televizijo na podlagi senzorskega sistema, ki je računal koordinate nahajanja roke na podlagi mikrofonov in zvočnikov, ki so oddajali ultrazvočne signale. Kljub dobri zamisli, sistem ni deloval zanesljivo in je bil težaven in neroden za upravljanje.

SLIKA 1-11: Nintendo Power Glove

Po letu 1990 je bilo še par poizkusov izdelave digitalne rokavice, vendar nobeden ni bil komercialno uspešen predvsem zaradi pomankanja programske opreme, ki bi omogočala izkoriščanje zmožnosti rokavice.

Navdušenje nad kinetičnimi vmesniki je leta 2006 povrnil Nintendo z konzolo Wii, ki je omogočala igranje iger z uporabo posebnega gamepada, ki je bil opremljen s senzorji za prepoznavo gibanja, pospeškov in orjentacije (slika1-12). Bil je velik prodajni uspeh in je navdušil konkurenčni podjetji Microsoft in Sony, da izkoristijo franšizo kinetičnih naprav.

SLIKA 1-12: Nintendo WiiRemote

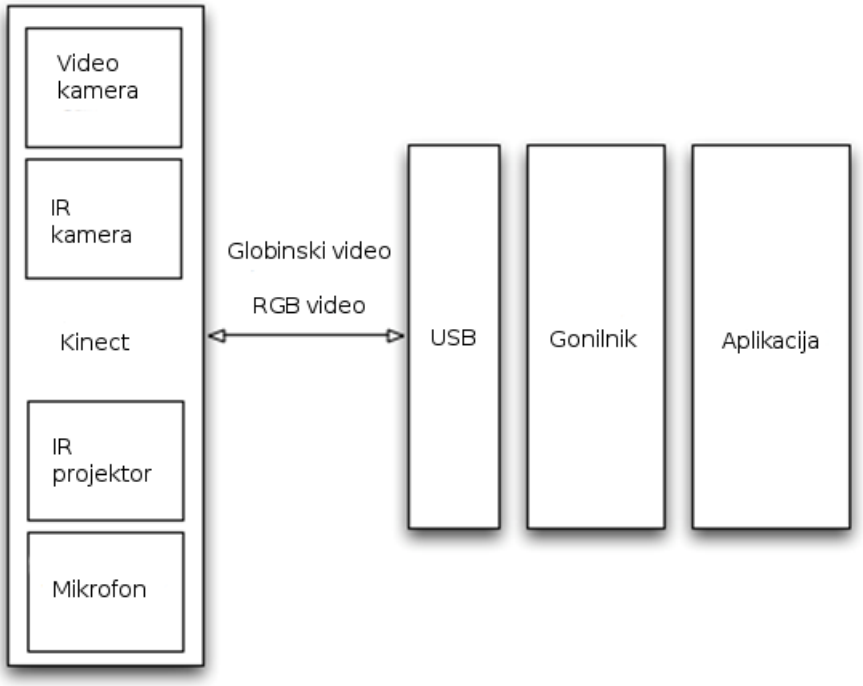
1.5 KAJ JE KINECT

Kinect (pred izidom bolj znan kot Project Natal) je igralni pripomoček za igralno konzolo Xbox 360 in osebne računalnike, ki poganjajo Windows 7 operacijski sistem. Njegov namen je igralca potopit v igro samo, z načelom da igralec nadzira igro le z gibi svojega telesa ali z govornimi ukazi, namesto, da bi zgolj pritiskal na gumbe (slika 1-13). Želji po drugačni igralni izkušnji potrjuje podatek, da je le v dveh mesecih po svojem izidu leta 2010 bilo prodanih že 8 miljonov enot. Dve leti kasneje je bila še izdelana verzija za Windows operacijske sisteme. Projekt Natal je bil zagnan leta 2005 in je plod dela Microsoftovega studija Rare [7], ki je razvijal programsko opremo in družbe PrimeSense [8], ki je razvila senzorski sistem. Slednji vključuje kamero, infra rdeči senzor globine in sistem mikrofonov. Kinect je bil prvič predstavljen javnosti leta 2009 na sejmu E3. Trenutno obstajata dve verziji Kinecta: za konzolo Xbox in PC. Različica za PC je precej posodobljena.

SLIKA 1-13: Kinect kot igralni pripomoček

1.5.1 KAKO NAPRAVA KINECT DELUJE

Senzorski sistem se deli na avdio in video del. Avdio del predstavljajo mikrofoni, preko katerih lahko sistemu dajemo govorne ukaze, oziroma ga lahko uporabljamo za glasovno komunikacijo z drugimi osebami preko spleta. Video del pa predstavlja kombinacija souporabe VGA kamere in 3D senzor globine, slednji je sestavljen iz infra rdečega oddajnika, in infra rdeče kamere. Kinect je z osebnim računalnikom povezan preko USB vodila (slika 1-14)



SLIKA 1-14: Diagram delovanja Kinecta

Oddajnik izžareva infra rdečo svetlobo v prostor, katera na objektih v vidnem polju ustvari vzorec (angl. spackle), ki je sestavljen iz krogcev. IR kamera nato pregleda vidno polje in primerja znani vzorec sestavljen iz krogcev z dobljenim vzorcem, na kateremu išče anomalije. Krogci se namreč z oddaljenostjo spremenijo v elipse, katere orientacija pomeni oddaljenosti. Tej tehniki pravimo strukturirana svetloba. Dodatna metoda za določanje globine je preprosto načelo, da objekti z odaljenostjo postanejo bolj zamegljeni.

Dobljene anomalije so obdelane v odločitvenih drevesih, ki so povezana v gozd. Po končanem postopku so humanoidne podobe ločene od ostalih anomalij, kot so recimo pohištvo, domače živali in vsakodnevni predmeti.

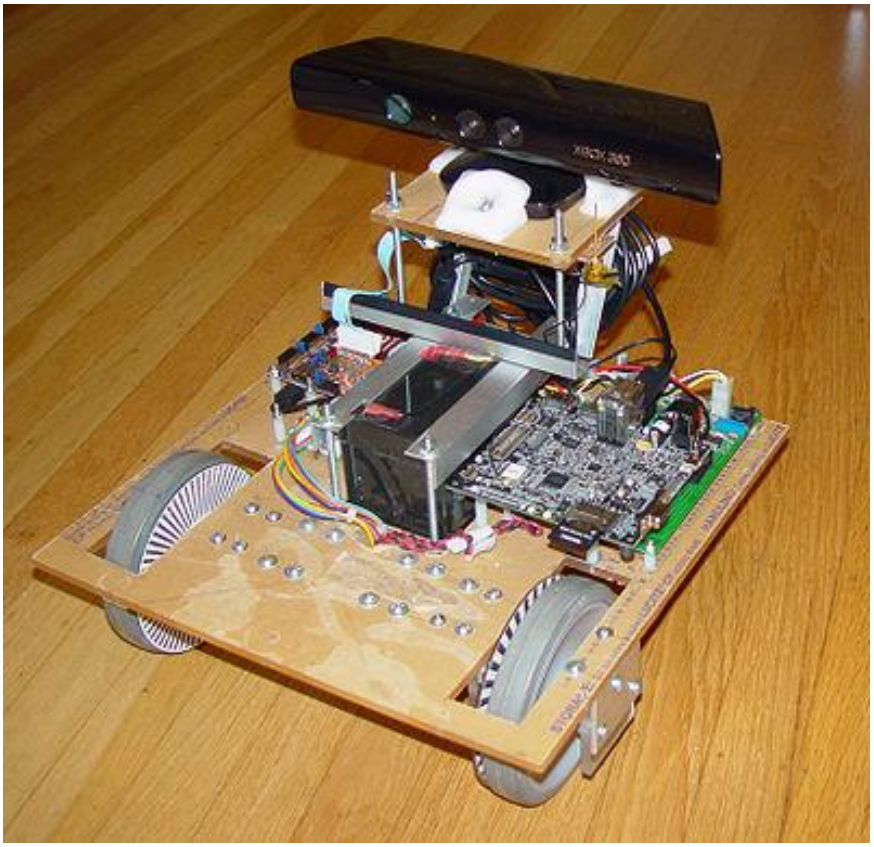
Dobljena podoba človeka se nato primerja z 100000 naučenimi vzorci slik z že določenimi skeleti. Na podlagi primerjav skozi odločitvena drevesa je dobljeni realni sliki določen skelet (slika 1-15).

VGA kamera pri zaznavanju gibov nima nobene vloge, saj opravlja delo navadne spletne kamere.

SLIKA 1-15: Zajeta slika z Kinectom in njeno procesiranje

1.5.2 KINECT SDK IN ODPRTOKODNE SOLUCIJE

Kinect ni zanimiv le zaradi svoje sposobnosti potopit igralca v igro, temveč tudi zaradi dejstva, da vsebuje inpresivno tehnologijo senzorskega sistema. Podobni sistemi lahko stanejo tudi več tisoč evrov, Microsoft pa prodaja Kinect pod ceno. Zaradi slednjega je postal precej zanimiv za širok spekter tehnično podkovanih ljudi: od domačih računalniških navdušencev pa vse do znanstvenikov, saj se Kinectove senzorje lahko uporabi kot vhodno napravo za računalnik, kot senzorski sistem v robotiki (slika 1-16) ali astronavtiki... Ob izidu je kup računalniških navdušencev tekmovalo, kdo bo prvi uspel povezati in uporabljati Kinect z osebnim računalnikom. Ponujene so bile celo nagrade v vrednosti par tisoč dolarjev. Dodatno vzpodbudo je dal Microsoft, ki je zatrdil, da ne bo pravno preganjal uporabnikov, ki hočejo uporabljati Kinect v druge namene, oziroma ga modificirati. Kinect je bil razbit v nekaj dneh od datuma izida. Kmalu za tem so bili napisani prvi neuradni gonilniki za operacijski sistem Linux, ki so omogočali uporabo kamere in infra rdečega senzorja. Decembra 2010 je podjetje PrimeSense, ki je sodelovalo pri nastanku Kinecta izdalo prve uradne gonilnike in programsko opremo za spremljanje gibanja. Na sredini leta 2011 je Microsoft izdal razvojno okolje za Kinect za sisteme z operacijskim sistemom Windows 7. Okolje vsebuje gonilnike za upravljanje Kinecta, dokumentacijo in programske primere aplikacij [9].



SLIKA 1-16: Kinect kot senzorski sistem robota

1.5.3 PRIMERJAVA MED KINECT SDK IN ODPRTOKODNIMI REŠITVAMI

Trenutno je na razploago možnih več razvojnih okolj. Sledi kratek opis vseh okolj z navedenimi prednostnimi in slabostmi.

- Microsoft Kinect SDK:

Kot je že iz imena razvidno je to uradno Microsoftovo razvojno okolje. Uporablja uradne gonilnike in znanje kako sistem deluje, kar omogoča stabilnost. Uporabnik ga lahko namesti brezplačno, vendar za njegovo uporabo potrebuje kopijo Microsoft Visual Studio 2010 in Windows 7 operacijski sistem. Dodatna omejitev je delitev Kinecta na Xbox in PC verzijo; uporaba Xbox verzije na PCju s Kinect SDK lahko povzroči omejitve pri izrabi polne funkcionalnosti in možnosti programiranja. Omejena je tudi izbira programskih jezikov: C++, C# , Visual Basic in JavaScript. Po drugi strani pa Microsoft nudi veliko dokumentacije in podpore. Omogočen je priklop in uporaba do 4 kinect naprav na enem PCju istočasno [10].

- Libfreenect:

Za libfreenect stoji spletna skupnost OpenKinect. Uporablja gonilnike, kateri so bili izpeljani iz postopka reverse engineeringa uradnih gonilnikov med tekmovanjem, kdo bo prvi prilagodil Kinect za delovanje na namiznem računalniku. Uporabnik ga lahko namesti brezplačno, ni potrebe po uporabi Visual Stuido 2010 ali omejitve na le Windows 7 operacijski sistem, saj deluje tudi na ostalih Windows izdajah. Linuxu in Mac OS X. Podprti programski jeziki so: C++, C#, C, Python, Java, JavaScript. Za dokumentacijo in podporo skrbi skupnost uporabnikov, ki v prostem času pomaga s svojim delom na projektu. Dodatno omogoča delo z motorji za usmirjanje Kinecta [11].

- OpenNI+SensorKinect:

Open Natural Interacion je neprofitna organizacija, katera se ukvarja z razvojem naravnih uporabniških vmesnikov. Med njenimi partnerji najdemo družbo PrimeSense, katera je razvijala Kinect. Slednja je javnosti izdala odprtokodni gonilnik za upravljanje Kinecta, ki deluje v Windows, Linux in Mac OS X operacijskih sistemih. Podprti programski jeziki so: C++, C#, C, Python, za dokumentacijo in podporo skrbi skupnost uporabnikov. OpenNI podpira tudi delo z Asus Xtrion Pro, napravo ki je zelo podobna Kinectu in jo izdeluje podjetje Asus [12].

- CL NUI platforma:

Za tem SDKjem stoji podjetje Code Laboratories, ki se ukvarja z razvojem in raziskavami realno časovnih interaktivnih sistemov. Uporablja odprtokodni gonilnik in podpira operacisjke sisteme Windows XP, Vista in 7. Podprti programski jeziki so: C# in C++. Omogočen je priklop in uporaba do 4 kinect naprav na enem PCju istočasno [13].

1.6 SORODNI UMETNIŠKI PROJEKTI REALIZIRANI S KINECTOM

Kinect je bil izdan novembra 2010, kmalu zatem so sledili še prvi neuradni gonilniki in programiranje aplikacij, katere izkoriščajo kinect se je lahko začelo. V obdobju skoraj dveh let je bilo izdelanih veliko aplikacij za različna področja uporabe. Večina jih je usmerjena na področje zabave (nadzor računalniških iger, multimedijskih vsebin) najdemo pa tudi aplikacije katere so namenjene asistenci zdravnikom in kirurgom v medicini, videonadzoru prostorov, aplikacije za spletne trgovine, predstavitve in alternativne umetniške aplikacije.

V tej diplomski nalogi bomo uporabili igralni pripomoček Kinect za navigacijo med prikazanimi umetniškimi deli. Osnovna funkcionalnost bo galerija slik, katero pa se lahko modularno razširi za prikaz drugih vrst vsebin, kot so: kipi, video posnetki ali animacije.

Pri tem velja omeniti slovensko podjetje Semantika d.o.o., ki se ukvarja z razvojem in implementacijo informacijskih rešitev. Pri iskanju inovativnih alternativ so se srečali tudi z napravo Kinect in za naročnike razvili aplikacije zanjo. Omeniti velja projekta prenove zelenih strani Zbornice za gospodarstvo [20] in postavitev info točke v avli podjetja Gen Energija [21]. Obe aplikaciji krmili uporabnik zgolj s kretnjami.

Sledi krajši seznam podobnih inštalacij na področju razstav in umetnosti:

- Digitalt! – knjižnica v Kopenhagnu (Danska)

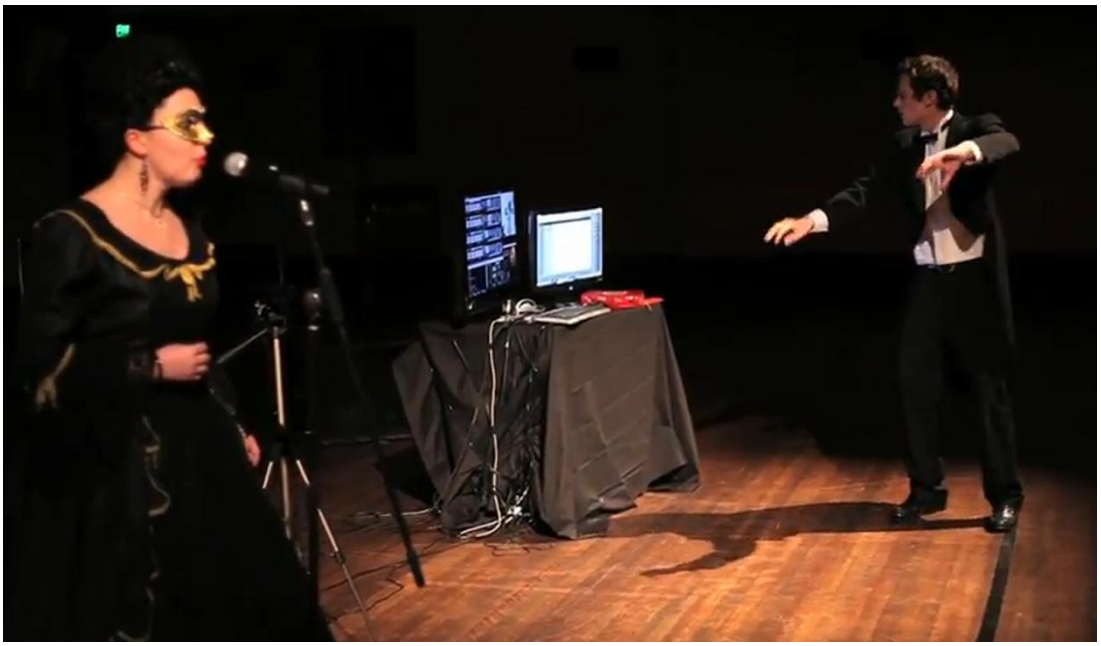
Namen te inštalacije je navidezna razširitev fizičnega prostora oziroma sobe. Sistem uporablja mrežo 3D zaslonov, ki tvorijo en velik zaslon in prikazujejo animirano ozadje, ki daje občutek globine. Ozadje se spreminja glede na položaj (kot) iz katerega si ga oseba ogleduje. Omogočena je tudi interakcija z ozadjem preko gibov (slika 1-17).

SLIKA 1-17: Interaktivni zid

Inštalacija temelji na Unity 3D grafičnemu pogonu in Kinectu za spremljanje položaja in gibov osebe, ki stoji pred zaslonom. Razvilo ga je Dansko podjetje YOKE, ki se ukvarja z razvojem uporabniških vmesnikov, senzorskih sistemov in aplikacij za računalniške sisteme, pametne telefone in tablične računalnike [14].

- Kinect kot klaviatura za orglje – Melburne (Avstralja)

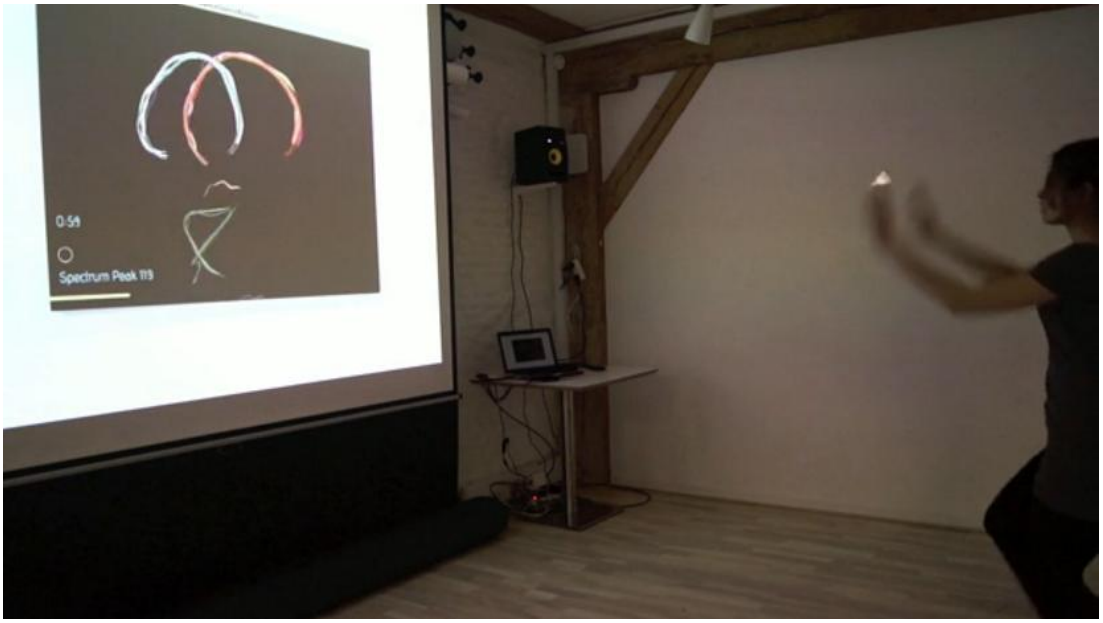
Chris Vik je leta 2011 razvil sistem, ki omogoča pretvorbo gibov rok v ukaze, katera nota bo bila igrana. V ta namen je napisal program Kinectar, ki zna MIDI napravam določati igrane note preko Kinecta (slika 1-18). Projekt je še vedno v fazi razvoja, testno verzijo pa je mogoče naložiti z uradne strani [15].



SLIKA 1-18: Koncert z uporabo Kinectarja

- Dance Dance Ribbon

Aplikacija je bila ustvarjena kot rezultat dvotedenskega tečaja, kjer so študentje imeli nalogo izdelati aplikacijo, preko katere bi se lahko umetniško izražali. Kot je že iz imena razvidno aplikacija preko Kinecta zajema gibanje plesalca in na zaslonu ustvarja igro raznobarvnih trakov. Dimenzije in intenziteta trakov so odvisne od hitrosti in načina premikanja rok in nog (slika 1-19). Podana je tudi možnost zajema slike vsake par sekund, kar omogoča plesalcu analizo in ogled svojega plesa in nastalih umetnin [16].



SLIKA 1-19: Aplikacija Dance dance ribbon med izvajanjem

- Kinect Turn – 3D lončarstvo

Turn je leta 2011 razvilo podjetje everyware in omogoča izdelavo 3D modelov skulptur z uporabo gest. Strojni del je sestavljen iz Kinecta, zaslona, računalnika z nameščeno aplikacijo in vrtljive lončarske mizice s senzorjem vrtenja. Uporabnik z vrtenjem mizice in premiki rok obdeluje digitalno glino v poljubno obliko, rezultat pa je viden na zaslonu (slika 1-20). Po končanem delu je mogoče 3D modele shraniti in jih fizično izdelati preko 3D tiskalnika [17].

SLIKA 1-20: Aplikacija Kinect turn med izvajanjem

2 RAZVOJ APLIKACIJE

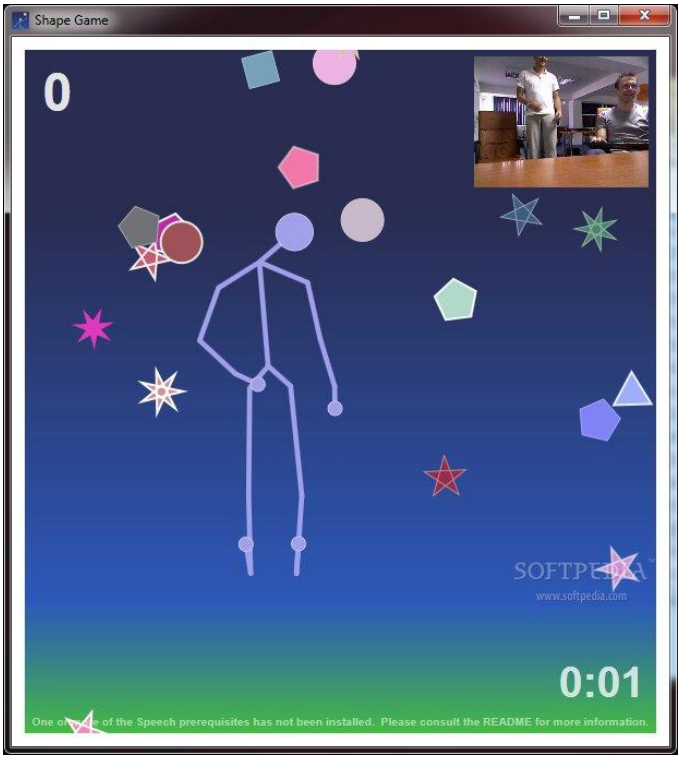
2.1 IZBIRA IN POSTAVITEV DELOVNEGA OKOLJA

Za razvoj in izdelavo aplikacije sem izbral Microsoft Kinect SDK 1.5, ki ponuja stabilno delovanje in dobro podporo. Poleg tega nam Fakulteta za računalništvo in informatiko nudi brezplačno uporabo Microsoftove programske opreme preko MSNDAA, tako da nismo imeli nobenih stroškov z nakupom operacijskega sistema Windows 7 Professional, kateri je bil uporabljen pri izdelavi diplomskega dela.

Postopek namestitve Kinect SDK je zelo enostaven:

- Z uradne Microsoftove strani se naloži in namesti usterzni Kinect SDK, bodisi 32 ali 64 bitno verzijo
- Po končani namestitvi sledi priklop Kinecta na napajanje in na USB vodilo
- Windows samodejno zazna Kinect in ustrezne gonilnike za njega. Kinect je takoj možno videti med napravami v upravitelju naprav

Celotna namestitev je opravljena v manj kot dvajestih minutah [10]. Uporabnik lahko začne nemudoma programirati ali se malo pozabava s priloženimi zgledi programov (slika 2-1).



SLIKA 2-1: Igra Shape game

2.2 IZBIRA IN OPIS IZBRANIH TEHNOLOGIJ ZA IZDELAVO APLIKACIJE

Za realizacijo galerije smo uporabili malo nenavaden programski jezik za tak projekt in sicer JavaScript. Posledično je ta aplikacija postala interaktivna spletna stran, ki se jo bodisi lahko poganja na lokalnem namiznem računalniku ali spletnem strežniku. Sledi kratek opis uporabljenih tehnologij:

- HTML

Kratica HTML pomeni Hyper Text Markup Lenguage oziroma jezik za označevanje nadbesedil. Uporaben je za izdelavo in prikazovanje spletnih strani v spletnem brskalniku. Ko odjemalec (uporabnik) pošlje zahteva za prikaz spletne strani strežniku, slednji odobri in pošlje nazaj HTML tekstovno datoteko, ki jo potem brskalnik pretvori in prikaže kot grafično spletno stran. HTML ima svoje začetke v zgodnjih devedesetih letih in je do danes doživel že veliko posodobitev. Zadnja posodobitev bo v verzijo 5, ki bo prinesla veliko sprememb.

- CSS

Cascading Style Sheets ali kaskadne stilske podloge so enostavni slogovni jezik, ki brskalniku povedo, kako in kje naj bi določeni html elementi bili postavljeni in kakšne lastnosti imajo. Cilj CSSa ni določanje novih pravil ali metod kako naj bi spletna stran izgledala, ampak je ločevanje stila od zgradbe strani. Na tak način osnovni HTML dokument postane veliko bolj pregleden, zmanjšano je ponavljanje kode in omogoča lažje spreminjanje izgleda ne da bi morali ponovno napisati celotni html dokument. Dela na projektu CSS so se začela leta 1994, prva verzija CSS 1 pa je izšla leta 1996.

- JavaScript

JavaScript je objektni skriptni programski jezik, ki omogoča izdelavo interaktivnih spletnih strani. Njegova sintaksa je zelo podobna Javi in Cju, toda z njima nima nič skupnega. Razvilo ga je podjetje Netscape, kot mikro operacijski sistem, ki bi lahko poganjal Javo in aplikacije napisanje v njej. Javnosti je bil predstavljen leta 1995 v spletnem brskalniku Netscape Navigator 2.0. Cilj Netscapa je bilo tekmovati z Microsoftom in njegovim Internet Explorerjem. Kasneje je tudi Microsoft podprl JavaScript in leta 1996 je bil JavaScript predlagan kot standardni programski jezik. Danes je JavaScript eden najbolj priljubljenih programskih jezikov med programerji.

2.2.1 KINECTJS

KinectJS je nastal iz ideje o uporabi Kinecta za upravljanje spletnih vsebin. Temelji na povezavi JavaScripta z naprednimi možnostmi HTML5. Za njegovo uporabo potrebujemo le Microsoftov Kinect SDK in spletni brskalnik Firefox ali Chrome na odjemalčevem računalniku. Na spletnemu strežniku pa je potrebno še namestiti aplikacijo KinectSocketServer (slika 2-2), ki je vez med gonilnikom za Kinect in JavaScript kodo.

Projekt je odprtokoden in je še v razvojni fazi. Dokumentacija je dosegljiva na uradni spletni strani [18].

Trenutne zmožnosti:

-omogoča podporo za dva uporabnika istočasno,

-spremljanje 20 sklepov, višino in dolžino okončin na posamezni skelet,

-spremljanje RGB slike in globinske slike,

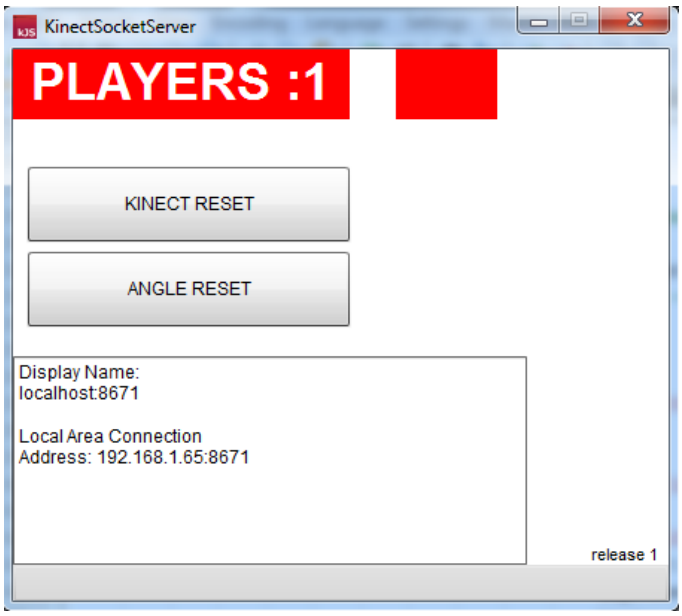
-nadzor motorja Kinecta,

-zajem slike preko kamere,

-preddefinirani gibi,

-možnost dodajanja drugih perifernih naprav, kot so mobilni telefoni,

-podpora za brskalnike Firefox in Chrome.



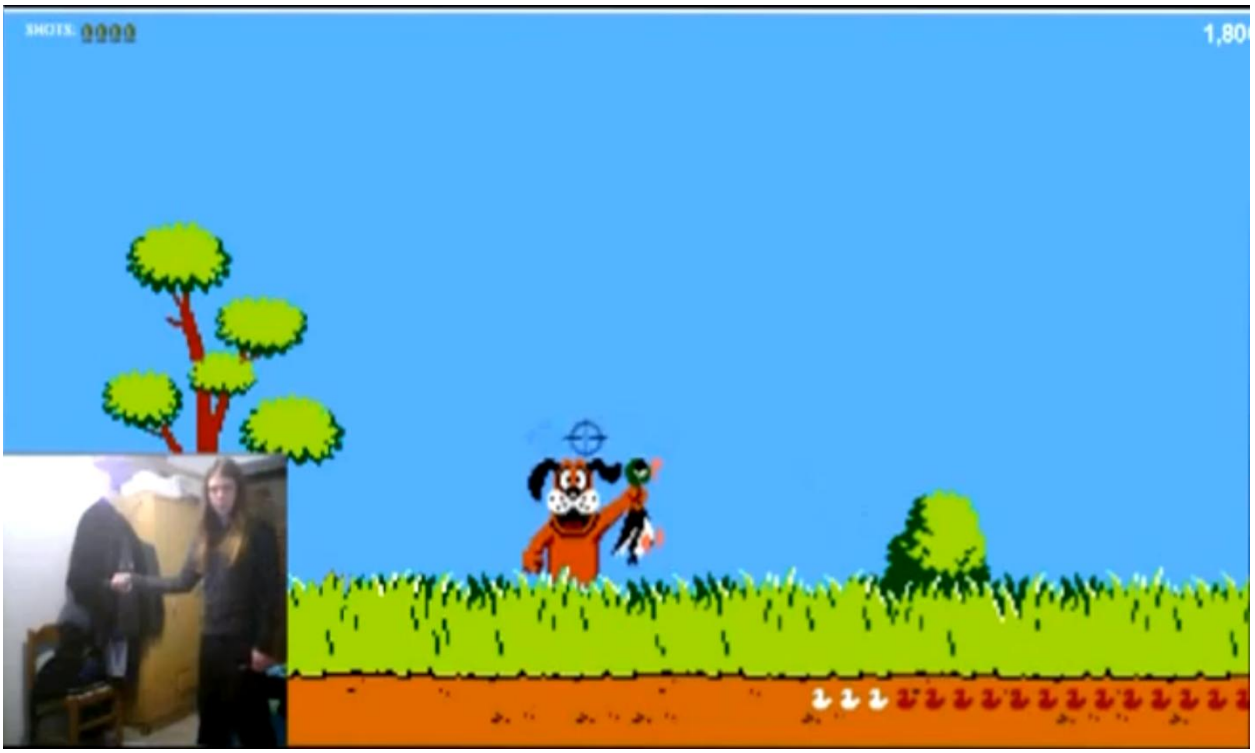
SLIKA 2-2: Aplikacija KinectSocketServer

2.2.1.1 PROJEKTI REALIZIRANI S KINECT.JS KNJIŽNICO

Prva izdaja kinect.js knjižnice je bila izdana 23.2.2012. Od takrat je bilo izdelanih par večjih projektov, ki prikažejo njene zmožnosti. Dosegljivi so na uradni strani knjižnice [18]. Sledi opis nekaterih bolj zanimivih projektov:

- Duckhunt

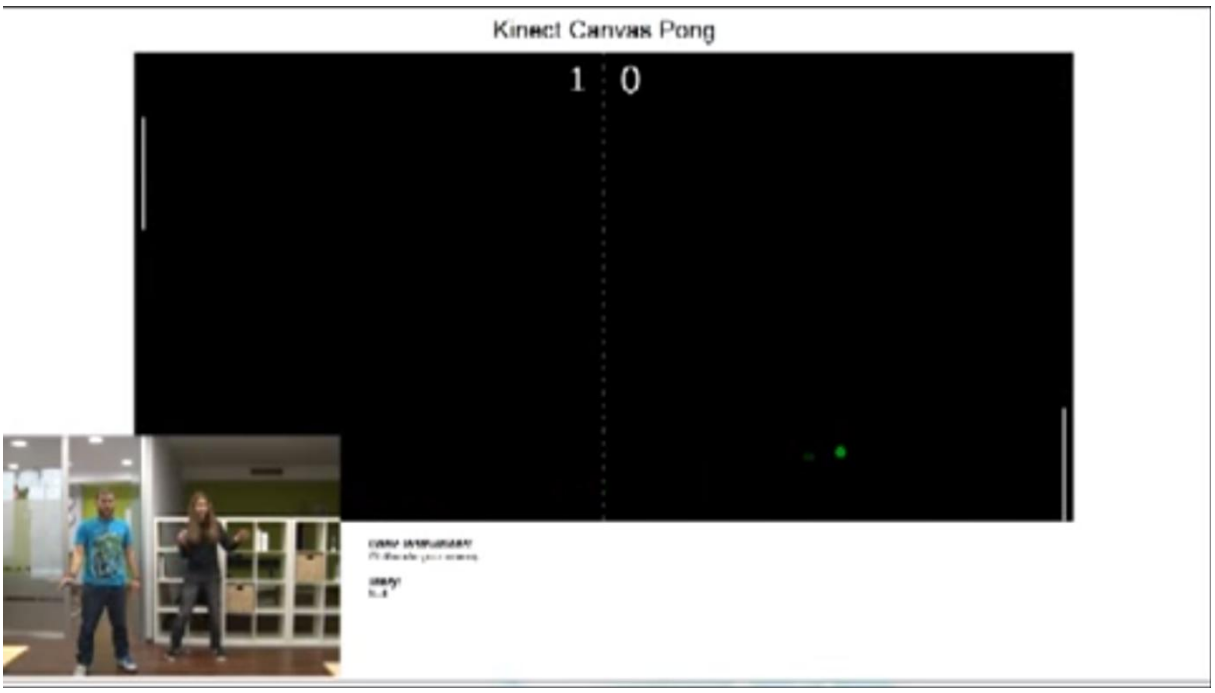
Gre za predelavo igre Duck Hunt iz leta 1984, katera je delovala na igralni konzoli Nintendo Entertainment System. Posodobljena različica je napisana zgolj v Javascriptu, CSSu in HTMLju (slika 2-3). Igralec s premiki roke premika merek na zaslonu in poiskuša zadeti raco. Ko je merek na raci se s pritiskom na gumb na mobilnem telefonu izvede strel.



SLIKA 2-3: Duckhunt

- Pong

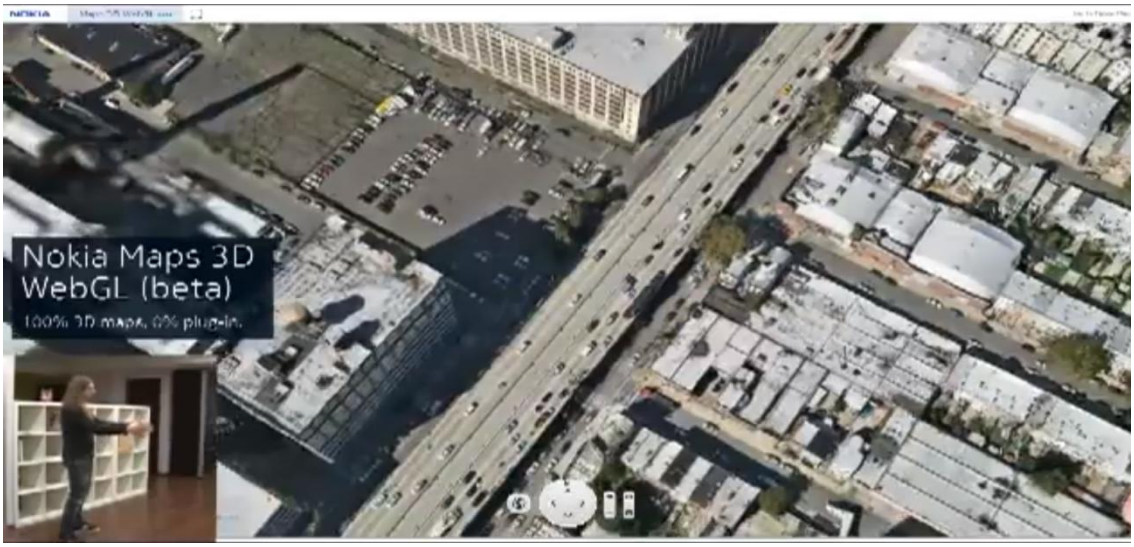
Pong, izdan leta 1972 je ena izmed prvih arkadnih iger, ki je omogočala igranje enostavne verzije tenisa. Izdelalo ga je podjetje Atari, ki je ravno zaradi priljubljenosti te igre šlo v razvoj iger in igralnih konzol. V njegovi posodobljeni različici lahko dva igralca istočasno s kretnjami pomikata vsak svoj plošček in z njima odbijata žogico in tekmujeta drug proti drugemu (slika 2-4).



SLIKA 2-4: Pong

- Nokia maps

Nokia maps (predhodno znan kot Ovi Maps) je Nokijin sistem zemljevidov za navigacijo na mobilnih napravah in namiznih računalnikih. Po zasnovi in funkcionalnosti je zelo podoben Google Maps. V projektu razvitim z Kinectom je omogočena manipulacija z zemljevidi preko kretenj uporabnika. V trenutni izdaji je mogočeno pomikanje po zemljevidu, približevanje in oddaljevanje zemljevida in spreminjanje načina pogleda iz satelitske slike v 3D izris terena in obratno (slika 2-5).



SLIKA 2-5: Nokia maps

- WebGL Cars

Cars je preprosta igra vožnje avtomobila narejena s pomočjo WebGl – JavaScript knjižnice namenjene za ustvarjanje 2D in 3D grafike. Pri tej igri igralec upravlja model avtomobila med vožnjo po prostoru z uporabo kretenj (slika 2-6).

SLIKA 2-6: WebGl Cars

2.3 APLIKACIJA KINECT GALERIJE

Namen aplikacije je povezati različne tehnologije za izdelavo preproste galerije slik, ki se jo lahko krmili bodisi s tipkovnico ali Kinectom. Možnosti upravljanja z govornimi ukazi ne bo bil uporabljen.

Pri razlagi se ne bomo osredotočali na vsako vrstico CSS, HTML ali JavaScript kode, temveč samo na tiste, katere so bistvene za delovanje galerije. Programiranje v JavaScriptu z knjižnico KinectJS ne zahteva nobenega posebnega programa za razvoj, saj zadošča že enostavna Beležnica. Za potrebe te naloge je bila izbrana sledeča programska oprema:

- Adobe Dreamweaver:

je aplikacija za razvoj spletnih strani. Zaradi njene zmogljivosti in enostavnosti je zelo pogosto uporabljena v krogih domačih in poklicnih izdelovalcem spletnih strani. V njej smo izdelali osnovni CSS in HTML spletne galerije.

- Notepad++

je enostavni tekstovni urejevalnik, ki nudi podporo za številne programske jezike. Velja za nepogrešljiv pripomoček pri programiranju. V računalniških krogih je dosegel status »švicarskega noža« zaradi svoje uporabnosti. Z njim smo programirali napredni JavaScript za Kincet.

Da bi prihranili na času, je bil uporabljen osnutek galerije Impress.js. Gre za enostavno ogrodje, ki omogoča izdelavo spletnih strani, galerij ali predstavitev. Napisal jo je Bartek Szopka in je dostopna preko GPL (general public lisence) za prenos s spleta [15].

Aplikacija se v grobem deli na dva dela: na samo galerijo (slika 2-7), kjer se lahko sprehajamo skozi seznam del, ter na podstrani, kjer si lahko ogledamo podrobnosti o določenem delu. Na podstraneh je omogočen tudi ogled 3D modelov, katere se lahko manipulira in obrača za 360°. Aplikacijo je v celoti mogoče uporabljati zgolj s pomočjo gibov.

Slika 2-7: Galerija

2.3.1 UVOZ KNJIŽNICE KINECT.JS

Preden začnemo programirati s KinectJS je potrebno uvoziti kinect.js knjižnico v html dokumentu. To lahko storimo na dva načina, ki sta odvisna od lokacije nahajanja kinect.js knjižnice. Lahko se nahaja lokalno [1] na spominskem mediju ali pa na spletu [2] (Programska koda 2-1). Za naš projekt dostopamo do lokalne shranjene kopije kinect.js, saj obstaja možnost, da naš računalnik ne bo stalno povezan na splet.

```
<html>
<head>
<!--(1)--><script type="text/javascript" src="../../../kinect.js"></script>
<!--(2)--><script type="text/javascript" src="http://kinect.childnodes.com/kinect.js"></script>
<script type="text/javascript">
    // javascript koda
</script>
</head>
<body>
    <!-- html koda -->
</body>
</html>
```

PROGRAMSKA KODA 2-1: Uvoz Kinectjs knjižnice

2.3.2 DOLOČANJE PARAMETROV IN SPREMENLJIVK

Z namenom poenostavitve programiranja Kinecta v JavaScriptu je programska koda precej podobna tisti za delo s tipkovnico. Imamo poslušalce (eventListnerje), kateri spremljajo če je uporabnik storil določene geste in jih pretvorijo v nadalne ukaze. Zaradi velike količine možnih gest je potrebno določiti parametre, katere bo Kinect spremljal. To storimo s klicem funkcije setUp() (Programska koda 2-2).

```
document.addEventListener( 'DOMContentLoaded', function() {
                    kinect.setUp({
                        players :   1,
                        relative    : true,
                        meters      : false,
                        sensitivity : 1.2,
                        joints  :   [ 'RIGHT_HAND', 'HAND_LEFT' ],
                        gestures:   [ 'SWIPE', 'ESCAPE' ]
                    })
                    .sessionPersist()
                    .modal.make();
```

PROGRAMSKA KODA 2-2: Določanje spremljanih parametrov

Število podanih parametrov je odvisno od končne funkcionalnosti aplikacije. Minimalno pa je potrebno definirati število uporabnikov, katere okončine spremljamo in kretnje. Seznam okončin in kretenj so že določene v naprej in ne moremo dodajati poljubnih.

- players (int): Število maksimalnih uporabnikov aplikacije (trenutno do 2)
- sensitivity (float): Nastavitev občutljivosti
- joints (array): Seznam okončin ali delov skeleta, katere spremljamo. Definiranih je več različnih delov skeleta: gležnji, komolci, noga, roka, bok, koleno, glava, rama, zapestje, nagib telesa... Dodatno je za vsaki del možno določiti levo ali desno stran. Celotni seznam se nahaja v knjižnici kinect.js [12].
- gestures (array): Seznam kretenj, ki jih spremljamo.

  Trenutno je definiranih več različnih kretenj (gestures), pri nekaterih pa je možno še določiti smer giba in okončino katera izvede gib, kar skupno nanese več možnih kombinacij:

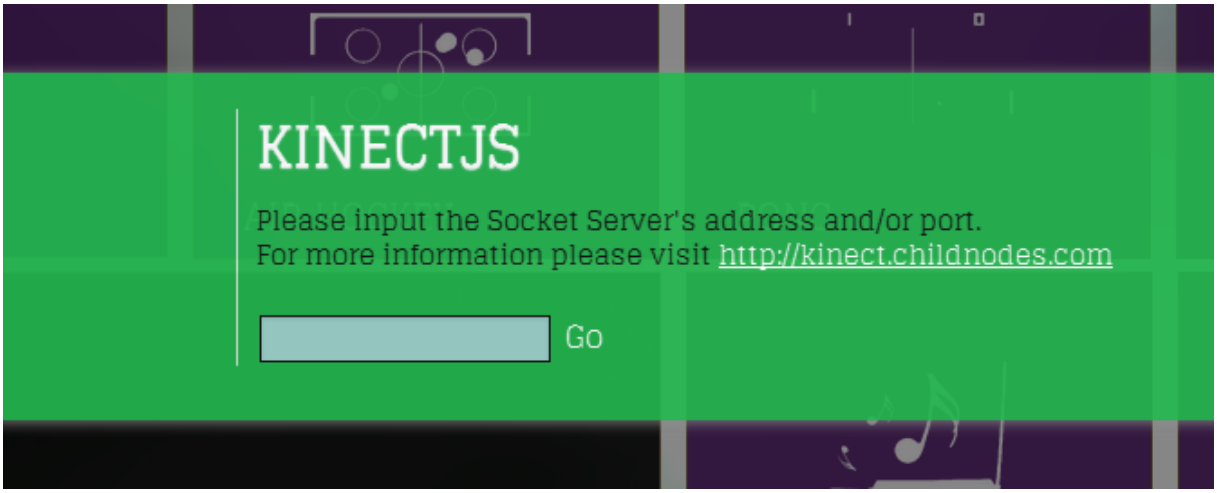
  - SWIPE: Zamah z roko, možno je definirati ali to storimo z levo ali desno roko in smer v katero zamahnemo. Možnosti so iz leve proti desni, iz desne proti levi, od spodaj gor in od zgoraj dol.
  - ESCAPE: Dvignjena roka, katera ostane v takem položaju za nekaj sekund
  - HANDS_DIST: Spremlja ali sta roki iztegnjeni ali pokrčeni pred uporabnikom. Glede na pozicijo rok se določi dogodke gestureCrank_ON (iztegnjene) in gesture Crank_OFF (pokrčene).
  - FOOT_LEAN: Zamah z nogo, možno je definirati ali to storimo z levo ali desno nogo in smer v kateri zamahnemo. Možnosti so iz leve proti desni, iz desne proti levi, od spredaj nazaj in od zadaj naprej.
  - JUMP: Spremlja ali je uporabnik skočil v zrak
  - BODY_ANGLE: Spremlja v katero smer je uporabnik nagnjen na področju ramen. Možna sta nagiba v levo ali desno smer.

S sessionPersist() pa dosežemo da seja (povezava z napravo) ostane živa tudi med navigacijo med podstranmi na isti domeni.

Parametre lahko naknadno spreminjamo tekom aplikacije s klicem funkcij:

- setPlayers (int),
- setSensitivity (float),
- setJoints (array),
- addJoint(string).

Pomembno vlogo ima tudi funkcija make(), ki za paramter vzame niz. Ta niz predstavlja IP naslov strežnika, kjer teče aplikacija KinectSocketServer. Če se izvaja lokalno mu je potrebno posredotavti le številko vrat. V trenutni verziji je potrebno to ročno vpisati v pojavitveno okence (slika 2-8). IP naslov najdemo v aplikaciji KinectSocketServer.



SLIKA 2-8: Okno za vnos naslova strežnika

Za enkrat je naslov potrebno ponovno vpisati ob vsakem ponovnem zagonu KinectSocketServer aplikacije. Z izdajo nove verzije in z izdajo izvorne kode pa bo to verjetno avtomatizirano, oziroma se bo pojavila kakšna aplikacija, katera bo znala samodejno posredovati naslov.

Nadaljno kodo za delo s Kinectom pišemo znotraj bloka document.addEventLsiter(){}. Kot je razvidno iz programske kode (programska koda 2-2) bomo spremljali samo levo in desno roko. Z desno roko roko se bomo pomikali skozi seznam del, bodisi naprej, bodisi nazaj, z levo roko pa bomo dostopali do podrobnejših informacij o delu in se vračali nazaj na seznam del.

2.3.3 Obvestila o statusu naprave in detekciji uporabnika

Vednost uporabnika kaj se dogaja s sistemom je ključnega pomena pri aplikacijah. To mišljenje je razvidno pri Nielsonovih principih, ki služijo za korektno načrtovanje uporabniških vmesnikov. Za omogočanje spremljanja obvestil je potrebno klicati funkcijo: notif.make();

Osredotočeni smo na dve kategoriji obvestil:

- obvestila o povezanosti naprave

Na sledeči način (programska koda 2-3) preverjamo ali lahko dostopamo do naprave Kinect iz JavaScript kode preko vtičnice (angl. Socket) s pomočjo aplikacije KinectSocketServer. Samo delovanje naprave je mogoče tudi fizično opaziti: v stanju pripravljenosti na Kinectu utripa zelena LED dioda, med delovanjem pa se leča IR projektorja obarva rdeče. Aplikacija KinectSocketServer ima tudi možnost samodejne ponastavitve kota naprave; in sicer če

uporabnik ni zaznan v vidnem polju, naprava spremeni naklon in s tem pregleda prostor za morebitnimi uporabniki.

```
61          //obvestila glede statusa sistema
62          kinect.addEventListener('openedSocket', function() { this.notif.push( "CONNECTED" ); });
63          kinect.addEventListener('closedSocket', function() { this.notif.push( "DISCONNECTED" ) });
```

PROGRAMSKA KODA 2-3: Obvestila o povezanosti naprave

- obvestilo o detekciji uporabnika

Glede na tehnologijo obstajajo določene omejitve za detekcijo uporabnika v Kinectovem vidnem polju. Za optimalno delovanje mora biti uporabnik oddaljen vsaj 1,8 metra od senzorja. Če nastopata dva uporabnika pa morata biti oddaljena vsaj 2,4 metra. Pomembna je tudi višina, kjer se senzor nahaja; optimalna je med 0,6 metra in 1,8 metra. Med upravljanjem je tudi pomembno, da med Kinectom in uporabnikom ni ovir, ter da v prostoru ni premočne svetlobe. Ali se uporabnik nahaja v vidnem polju preverimo na sledeči način (programska koda 2-4).

```
65          //obvestila o najdenemu ali izgubljenemu igralcu/uporabniku
66          kinect.addEventListener('playerFound', function( count ) {
67              this.notif.push( "PLAYER FOUND. Total : " + count[ 0 ] )
68          });
69
70          kinect.addEventListener('playerLost', function( count ) {
71              this.notif.push( "PLAYER LOST. Total : " + count[ 0 ] )
72          });
```

PROGRAMSKA KODA 2-4: Obvestila o detekciji uporabnika

Detekcijo uporabnika se lahko v našem primeru izrabi v dodatno funkcionalnost galerije. Če naprava ne zazna uporabnika v njenem vidnem polju se na zaslonu umetniška dela samodejno pomikajo skozi seznam. V poslušalcu spremljamo ali je oseba zaznana. V primeru da je ni se izvede funkcija setInterval(), katera vsake 10 sekund kliče funkcijo next() ali prev(), ki povzroči pomik (programska koda 2-5). Spremenljivka mesto služi za določanje na kateri strani se nahajamo in s tem možnost za ogled dodatnih informacij o delu.

```
kinect.addEventListener( 'playerLost', function( count ) {
    if( count[ 0 ] === 0 ){

        if(mesto < 3000){
            setInterval(impress().next();, 10000 );
            mesto = mesto + 1000;
        }
        if(mesto > 0){
            setInterval(impress().prev();, 10000 );
            mesto = mesto - 1000;
        }

    }
    else
        return false;
});
```

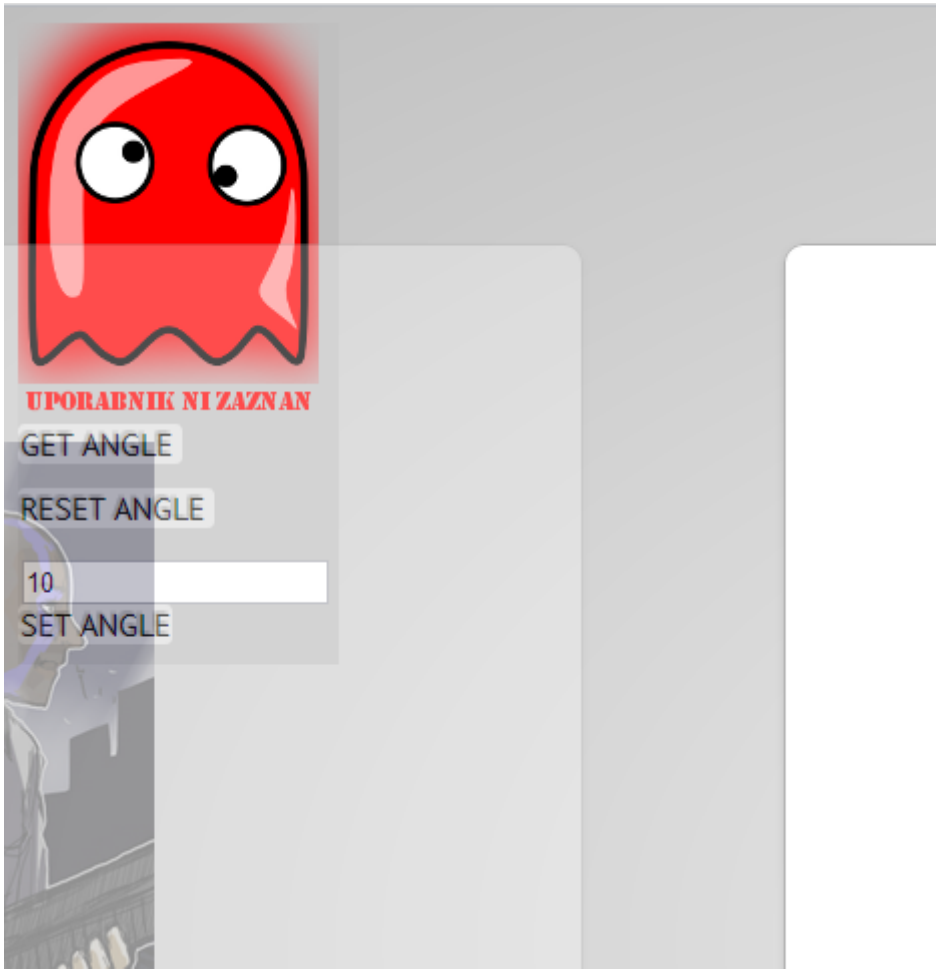
PROGRAMSKA KODA 2-5: Samodejni pomik

2.3.4 OKNO ZA SPREMLJANJE STATUSA IN UPRAVLJANJE Z MOTORJEM NAPRAVE

Za pravilno delovanje Kinecta, mora biti uporabnik v njegovem vidnem polju. Aplikacija KinectSocketServer nam zna sporočiti ali je zaznala uporabnika in samodejno nastaviti kot naprave, vendar želimo imeti isto funkcionalnost tudi znotraj galerije. V ta namen smo postavili ločeni razred v zgornji levi kot (programska koda 2-6). V njem bo prikazan status o zaznavi uporabnika ter možnost prikazati, nastaviti in ponastaviti kot Kinect naprave (slika 2-9).

```
195 <div style="width:160px; height:320px; background-color: #C0C0C0;">
196
197     <div id="skCount"></h4><img id="skImg" src="ikone/nizaznan.png" /><span></span></div>
198
199     <div id="controls">
200             <a href="#" id="getAngle"> GET ANGLE </a> <br /><br />
201             <a href="#" id="resetAngle"> RESET ANGLE </a> <br /><br />
202             <input id="deg" type="text" value="10" />
203             <a href="#" id="setAngle">SET ANGLE</a>
204     </div>
205
206
207
208
209 </div>
```

PROGRAMSKA KODA 2-6: HTML koda oken z statusom



SLIKA 2-9: Izgled okna z statusom

Ko uporabnika ni v vidnem polju, je prikazana ikona rdečega duhca, ko pa uporabnik stopi v vidno polje se slednji obarva v zeleno. Ob primeru izginotja uporabnika se ikona zamenja nazaj v rdečo (programska koda 2-7).

```
//detekcija uporabnika
//zaznan
kinect.addEventListener( 'playerFound', function( count ) {
    document.getElementById('skCount').innerHTML = count[ 0 ];
    if(  count[ 0 ] === 1 )     //zamenjava ikone
        document.getElementById('skImg').src = 'ikone/zaznan.png';

});
//izgubljen
kinect.addEventListener( 'playerLost', function( count ) {
    document.getElementById('skCount').innerHTML = count[ 0 ];

    if( count[ 0 ] === 0 )      //zamenjava ikone
        document.getElementById('skImg').src = 'ikone/nizaznan.png';
    else ( count[ 0 ] === 1 )
        document.getElementById('skImg').src = 'ikone/zaznan.png';

});
```

PROGRAMSKA KODA 2-7: Detekcija uporabnika

Delo z motorjem je enostavno, saj že obstajajo definirane funkcije za upravljanje in nastavljenje parametrov:

- getCurrentAngle()
  Funkcija dobi trenutno vrednost kota naklona naprave (programska koda 2-8).

```
document.getElementById('getAngle').addEventListener('click', function( e ) {
    e.preventDefault();

    kinect.motor.getCurrentAngle();
}, false);
```

PROGRAMSKA KODA 2-8: Funkcija, ki ugotovi kot naprave

- setCurrentAngle()
  Funkcija služi za nastavitev kota naprave. Razpon sega od -27°do +27° (programska koda 2-9).

```
document.getElementById('setAngle').addEventListener('click', function( e ) {
    e.preventDefault();

    kinect.motor.setCurrentAngle( document.getElementById('deg').value );
}, false);
```

PROGRAMSKA KODA 2-9: Funkcija za nastavljanje kota naprave

- defaultAngle()
  Funkcija povrne kot naprave na privzeto vrednost -5° (programska koda 2-10).

```
document.getElementById('resetAngle').addEventListener('click', function( e ) {
    e.preventDefault();

    kinect.motor.defaultAngle();
}, false);
```

PROGRAMSKA KODA 2-10: Koda za ponastavitev kota naprave

2.3.5 POMIKANJE SKOZI SEZNAM DEL

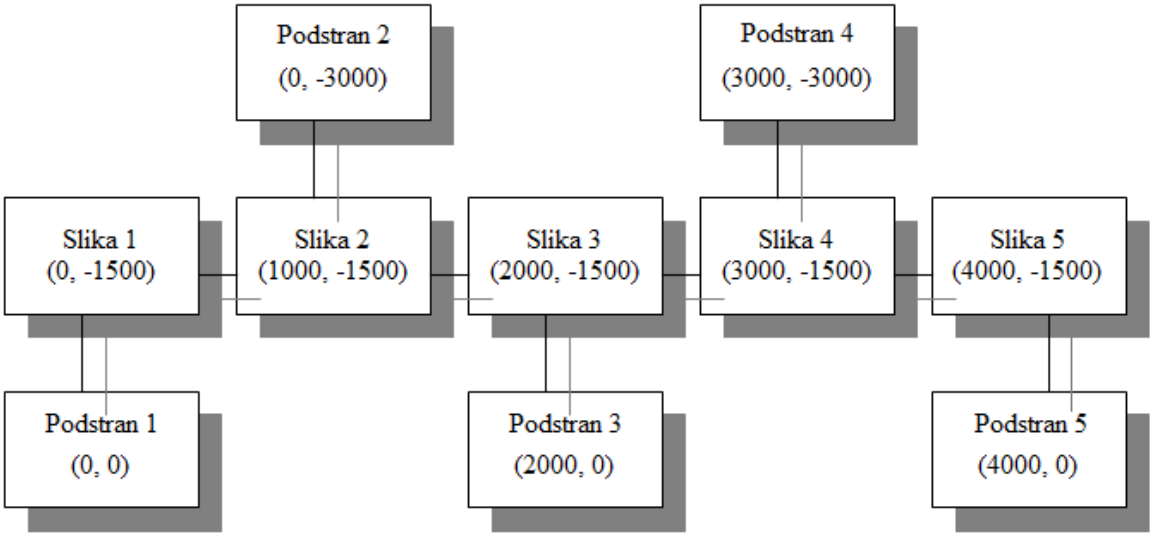
Kot je razvidno iz zaslonskega posnetka galerije (slika 2.7) je prikaz osredotočen le na eno delo naenkrat in možnost predogleda prejšnjega in naslednjega dela za lažjo orjentacijo. Uporabnik se lahko premika med deli na podoben način kot bi to storil v programih za predstavitve, kot so: PowerPoint, Keynote ali LibreOffice Impress.

Posamezna prosojnica je v bistvu ločeni razred z v njej definirano tabelo v katerih celicah so prikazane posamene slike oziroma podatki. Za tako strukturo smo se odločili zaradi poenostavitve strani, saj ni bilo potrebno pisati nove CSS kode s katero bi določili izgled in lego elementov (programska koda 2-11).

```
238    <div class="step slide" id="1" data-x="0" data-y="-1500">
239
240        <table width="780" height="600" border="0">
241          <tr>
242            <td height="15" colspan="5" align="center">Janez Novak - Left 4 Dead - 2009</td>
243          </tr>
244          <tr>
245            <td width="100" height="500"> </td>
246            <td colspan="3" align="center"><img src="slike/slika1.jpg" width="479px" height="434px"/></td> <!-- main body -->
247            <td width="100"> </td>
248          </tr>
249          <tr align="center">
250            <td height="100" width="100"><img src="slike/levo.jpg" alt="LEVO" /></td>
251            <td width="193"><img src="slike/slika4.jpg" width="193px" height="100px"/></td>
252            <td width="193">
253                <table border="1" bordercolor="#FF0000">
254                <tr><td><img src="slike/slika1.jpg" width="193px" height="100px"/></tr></td>
255                </table>
256            </td>
257            <td width="193"><img src="slike/slika2.jpg" width="193px" height="100px"/></td>
258
259            <td><img src="slike/desno.jpg" alt="DESNO" /></td>
260          </tr>
261        </table>
262
263
264    </div>
```

PROGRAMSKA KODA 2-11: HTML tabela

Pomik med stranmi omogoča funkcija, katera kot parameter prejme dodatne lastnosti *data-x* in *data-y* v znački . Če imata vrednosti *data-x*=-1000 in *data-y*=-1500, pomeni, da se sredina nahaja na koordinatah x=-1000 in y=-1500 na našem predstavitvenem platnu. Korak med podstranmi znaša x=1000 (slika 2-10) . Ko podamo ukaz za pomik na naslednjo ali prejšnjo prosojnico se »kamera« pomakne na območje kjer se nahajajo koordinate naslednika ali predhodnika. Velikost in merilo platna, okna in koraka pri tranziciji se prilagaja ločljivosti zaslona in velikosti okna spletnega brskalnika. Zaradi možnosti bolj naravnega izgleda med tranzicijo, je tudi mogoče določiti trajanje pomika.



SLIKA 2-10: Diagram prehajanja

Glavno delo pri pomikanju skozi seznam pa opravljaja poslušalec (angl. Event Listner). Slednji se neprestano izvajaja in spremlja če uporabnik z desno roko izvede zamah. Ko je taka kretnja zaznana se najprej preveri v kateri smeri je bila kretnja izvedena. Na podlagi tega se izvede koda znotraj ustreznega stikala (programska koda 2-12).

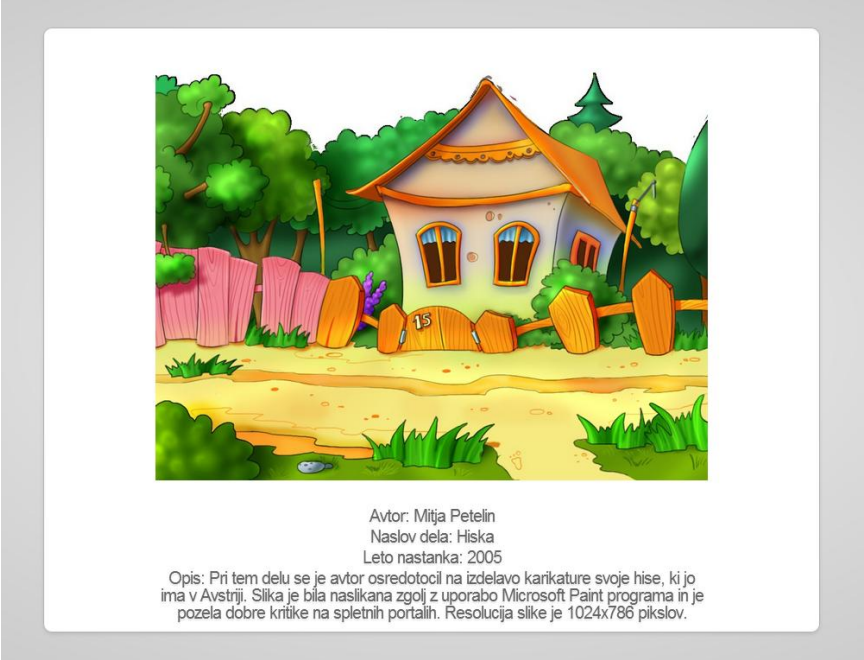
```
kinect.addEventListener( 'gestureSwipe', function( args ) {
    //args = array, 0 --> player index, 1--> joint, 2--> direction

    if( args[ 0 ] !== 0 )
        return false;

    if( args[ 1 ] === 11 )
    {
        switch( args[ 2 ] ) {
            case( 'left' ) :
                if(mesto < 4){
                    impress().next();
                    mesto = mesto + 1;
                }
            break;
            case( 'right' ) :
                if(mesto > 0){
                    impress().prev();
                    mesto = mesto - 1;
                }
            break;
        }
    }
});
```

PROGRAMSKA KODA 2-12: Poslušalec za pomik med stranmi

Kot je razvidno (programska koda 2-12) se poleg klica funkcije next() in prev() beleži tudi trenutna lokacija na kateri strani se nahajamo v spremenljivki *mesto*, katero povečujemo ali zmanjšujemo za vrednost 1, saj tako posplošimo označevanje strani. V odločitvenih stavkih spremljamo velikost spremeljivke *mesto*, da preprečimo obhod seznama. Spremeljivka mesto nam tudi služi za določitev katera je prava podstran z dodatnimi informacijami o delu.

2.3.6 PRIKAZ DODATNIH INFORMACIJ O DELU

Kot je razvidno iz galerije (slika 2-3) uporabnik dobi zelo malo informacij o delu. Za bolj podroben opis umetnine si lahko uporabnik pogleda podstran o njej (slika 2-11) z uporabo kretnje. Ko končna z ogledom se lahko z uporabo iste kretnje vrne na seznam del.



SLIKA 2-11: Podstran galerije

Za ta namen imamo poslušalca, ki spremlja ali je uporabnik dvignil levo roko. Ko zazna to kretnjo pogleda spremeljivko *mesto* in spremenljivko *flag*, katera nam pove ali se nahajamo na seznamu del ali podstrani. Na podlagi njunih vrednosti se postavi kamero na ustrezne koordinate.

```
kinect.addEventListener( 'gestureEscape', function( count ) {
    if(count[ 0 ] !== 0)
        return false;

    if(count[ 1 ] === true)
    {
        if(mesto == 0){
            if(flag==0){
                flag=1;
                window.location = "index.html#/6"
            }else{
                flag=0;
                window.location = "index.html#/1"
            }

        }
```

PROGRAMSKA KODA 2-13: Poslušalec za skok na pod stran

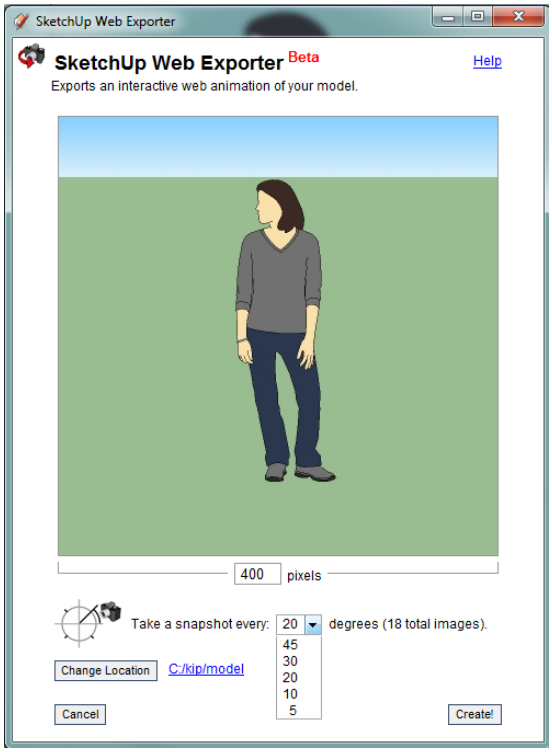
Če se nahajamo na seznamu del (*flag*=0) nas preusmeri na podstran o delu, če pa se nahajamo na seznamu del (*flag*=1) nas preusmeri na seznam del (programska koda 2-13).

2.3.7 PRIKAZ 3D MODELA KIPA

Poleg prikaza slik je bil omogočen še ogled skulptur. Zanje je značilna možnost ogleda iz vseh smeri, zato smo v aplikaciji omogočili uporabniku da z gestami poljubno obrača in si ogleduje skulpturo.

Prvotna zamisel je bila uporaba multimedijske platforme Adobe Flash, vendar ta še ne podpira direktne manipulacije s 3D modeli, oziroma je potrebno uporabiti program Swift3D, kateri pretvori modele v Flash format. Dobljeni rezultat ni pravi 3D model, ampak bolj animacija modela, kateri je bil slikan z vseh strani. Poleg tega Flash zahteva dodatno procesno moč, kar poveča stroške v primeru nakupa novega namenskega sistema za poganjanje naše aplikacije na dogodkih.

Končna rešitev uporablja podoben koncept slikanja 3D modela. S programom Google SketchUp smo ustvarili poljubni 3D model in ga z uporabo funkcije SketchUp Web Exporter slikali v krogu 360° (slika 2-12). Dobljene slike smo shranili v JavaScript tabelo. Za naš testni primer smo model slikali vsake 20° in ustvarili 18 slik. Za občutek zveznosti je priporočljiva velika gostota slik. Ta tehnika se imenuje 2.5D (dva in pol dimenzionalno) ali pseudo 3D, saj daje lažni občutek dimenzije. Uporabljena je predvsem v računalniških igrah za optimizacijo delovanja.



SLIKA 2-12: SketchUp web Exporter

Za preklapljanje med posameznimi slikami potrebujemo dve funkciji; menjajNaprej() in menjajNazaj(). Pri obeh imamo skupno globalno spremenljivko x, ki je na začetku nastavljena na 0. Pri klicu funkcije menjajNaprej() (programska koda 2-14) se slednja poveča za 1 in hkrati spremljamo da ne postane večja kot 17, saj bi tako dali referenco na napačno poimenovano sliko. V takem primeru ponastavimo x nazaj na vrednost 0, saj je bil opravljen obhod v nasprotni smeri urinega kazalca (programska koda 2-15).

```
10  function menjajNaprej(){
11      document.getElementById("kip").src=slike[++x];
12      if (x==17) {
13          x=0;
14      }
15  }
```

PROGRAMSKA KODA 2-14: Funkcija menjajNaprej()

Na podoben način deluje funkcija menjajNazaj() (programska koda 2-7), kjer pa z vsakim njenim klicem zmanšujemo vrednost x za 1. Ko pridemo do negativne vrednosti, spremenljivko x nastavimo na vrednost 17, saj bi drugače dali referenco na napačno poimenovano sliko.

```
17  function menjajNazaj(){
18      document.getElementById("kip").src=slike[x--];
19      if (x==-1) {
20          x=17;
21      }
22  }
```

PROGRAMSKA KODA 2-15: funkcija menjajNazaj()

Obe funkciji nato kličemo znotraj eventListner() (programska koda 2-16) glede na smer giba (zamaha) z roko. Gib zamah z roko je bil izbran zaradi poenotenja ukazov.

```
36          kinect.addEventListener( 'gestureSwipe', function( args ) {
37
38              if( args[ 0 ] !== 0 )
39                  return false;
40
41                  if( args[ 1 ] === 11 ) // smer
42                  {
43                      switch( args[ 2 ] ) {
44                          case( 'left' ) :
45                              menjajNaprej();
46                          break;
47
48                          case( 'right' ) :
49                              menjajNazaj();
50                          break;
51                      }
52                  }
53          });
```

PROGRAMSKA KODA 2-16: eventListner za rotacijo modela

2.4 UPORABA APLIKACIJE

Sama aplikacija je zelo enostavna za uporabo. Krmiljenje deluje skozi preddefinirane kretnje. Sledi kratek opis le-teh:

- Dodatne informacije o sliki / vrnitev na seznam del:

  Uporabnik iztegne levo roko v obliki črke L (slika 2-13).

SLIKA 2-13: dvig roke

- Pomik med deli levo / rotacija kipa v levo smer

  Uporabnik iztegne desno roko in z njo zamahne desno (slika 2-14).

SLIKA 2-14: Zamah desno

- Pomik med deli desno/ rotacija kipa v desno smer

  Uporabnik iztene desno roko in z njo zamahne levo (slika 2-15).

SLIKA 2-15: zamah levo

2.4.1 DODAJANJE NOVIH DEL V GALERIJO

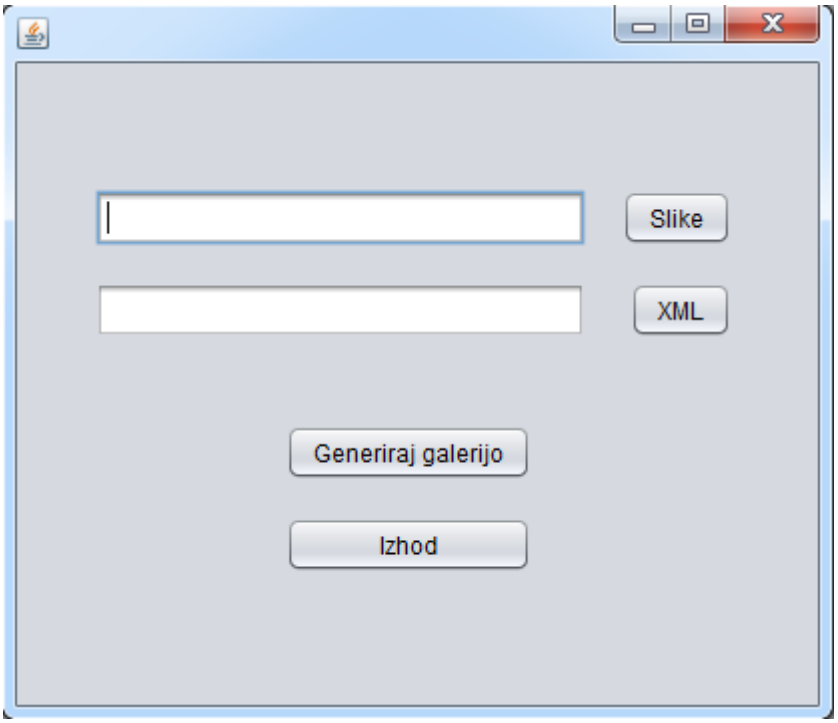
Trenutno je omogočeno dodajanje novih del v galerijo le preko pisanja nove HTML kode, kar je lahko ovira za neuke uporabnike in umetnike, ki bi želeli postavit lastno galerijo. Rešitev tega je ločena aplikacija napisana v poljubnem programskem jeziku, ki omogoča pisanje in branje datotek.

Za ta namen smo v programskem jeziku Java napisali aplikacijo, ki kot parametre dobi datoteko .xml (programska koda 2-17) s podatki o delih in mapo kjer so shranjena dela, katera želimo prikazati. Ob zagonu nato aplikacija ustvari datoteko .html z dinamično generirano vsebino. Takšna zasnova se izkaže za zelo učinkovito pri organizaciji večjega števila del.

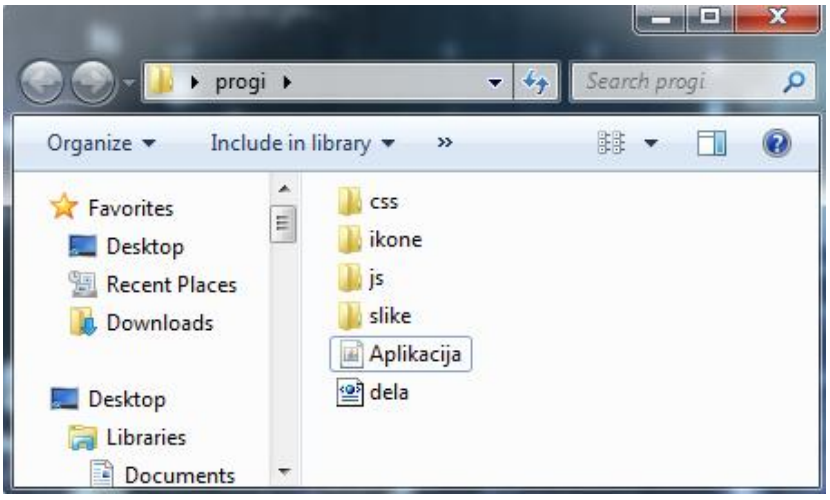
```xml
<?xml version="1.0"?>
<umetnine>
    <slika>
        <avtor>Janez Novak</avtor>
        <naslov>Left4Dead</naslov>
        <leto>2004</leto>
        <opis>opis slika1</opis>
    </slika>
    <slika>
        <avtor>Micka Novak</avtor>
        <naslov>Dr. House</naslov>
        <leto>2007</leto>
        <opis>opis slika2</opis>
    </slika>
    <slika>
        <avtor>Tretja Oseba</avtor>
        <naslov>Zelena luna</naslov>
        <leto>1999</leto>
        <opis>opis slika3</opis>
    </slika>
    <slika>
        <avtor>Mitja Petelin</avtor>
        <naslov>Hiska</naslov>
        <leto>2005</leto>
        <opis>opis slika4</opis>
    </slika>
</umetnine>
```

PROGRAMSKA KODA 2-17: XML

Za delovanje aplikacije (slika 2-16) je potrebno imeti nameščeno Java JDK. Aplikacijo poženemo z dvoklikom na ikono Aplikacija.jar in nato v njej preko izbirnika datotek (angl. File chooser) določimo željeno xml datoteko in mapo z slikami. Z klikom na gumb *Generiraj galerijo* aplikacija ustvari html datoteko galerije. Slednjo je potrebno kopirati v mapo, kjer se nahajajo mape s slikami, ikonami, css in statičnojavascript kodo (slika 2-17).



SLIKA 2-16: Aplikacija za tvorjenje galerije



SLIKA 2-17: Postavitev v mapi

2.4.1.1 PROGRAMSKA KODA APLIKACIJE

Na začetku aplikacije postavimo razčlenjevalnik DOM (programska koda 2-18), kateri iz .xml dokumenta zgradi drevesno strukturo.

```
36      DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();
37      DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
38      Document doc = docBuilder.parse (new File("dela.xml"));
```

PROGRAMSKA KODA 2-18: BuilderFactory

V zanki (programska koda 2-19) se nato sprehodimo čez mapo, kjer so shranjene dela. Sproti še preverjamo ali so vsebovane datoteke slike v formatu .jpg. Pri tem se za vsako sliko spremenljivka *stSlik* poveča za 1.

```
44      for (int i = 0; i < listOfFiles.length; i++)
45      {
46
47          if (listOfFiles[i].isFile())
48          {
49              files = listOfFiles[i].getName();
50              if (files.endsWith(".jpg") || files.endsWith(".JPG"))
51              {
52              stSlik++;
53              }
54          }
55      }
```

PROGRAMSKA KODA 2-19: stetje slikovnih datotek

V naslednjem koraku preverimo ali se število slik v mapi ujema s številom del v datoteki .xml (programska koda 2-20). Ob neujemanju se program prekine in izpiše napako.

```
57      NodeList listOfDel = doc.getElementsByTagName("slika");
58      int stDel = listOfDel.getLength();
59
60      if((stSlik+1) == stDel){
61          System.out.println("Napaka, st slik se ne ujema s seznamom del");
62      }
63      else{
```

PROGRAMSKA KODA 2-20: PReverjanje Števila del

Ob ujemanju program začne pisati v datoteko index.html statično (programska koda 2-21) in dinamično (programska koda 2-22) kodo HTML in JavaScript. Seznam del in podstrani se tvori glede na podlagi števila slik v datoteki.

```
63          else{
64
65          try {
66              FileWriter fstram = new FileWriter("index.html");
67              BufferedWriter out = new BufferedWriter(fstram);
68
69
70              out.write("<!doctype html>\n");
71              out.write("\n");
72
73              out.write("<html lang=\"en\">\n");
74              out.write("<head>\n");
75
76              out.write("\t<meta charset=\"utf-8\" />\n");
```

PROGRAMSKA KODA 2-21: pisanje v .html datoteko

```
177             for(int j=0; j < stSlik; j++){
178
179                 Node firstPersonNode = listOfPersons.item(j);
180                 if(firstPersonNode.getNodeType() == Node.ELEMENT_NODE){
181
182                     Element firstPersonElement = (Element)firstPersonNode;
183
184                     if(j==0){
185                         prejsnja=listOfFiles[stSlik-1].getName();
186                     }else{
187                         prejsnja=listOfFiles[j-1].getName();
188                     }
189
190                     if(j==stSlik-1){
191                         naslednja=listOfFiles[0].getName();
192                     }else{
193                         naslednja=listOfFiles[j+1].getName();
194                     }
```

PROGRAMSKA KODA 2-22: Računanje prejšnje in naslednje slike

Podatke kot so: naslov dela, avtor, leto nastanka in opis dela preberemo iz .xml datoteke (programska koda 2-23). Pri tem je nujno, da so slike v mapi urejene v istem vrstnem redu kot v .xml datoteki.

```
197         out.write("<div class=\"step slide\" id=\""+(j+1)+"\" data-x=\""+(j*1000)+"\" data-y=\"-1500\">  \n");
198         out.write("\n");
199         out.write("<table width=\"780\" height=\"600\" border=\"0\">\n");
200         out.write("<tr>\n");
201 
202         NodeList avtorList = firstPersonElement.getElementsByTagName("avtor");
203         Element avtorElement = (Element)avtorList.item(0);
204         NodeList textAvtorList = avtorElement.getChildNodes();
205         out.write("<td height=\"15\" colspan=\"5\" align=\"center\">"+((Node)textAvtorList.item(0)).getNodeValue().trim());
206 
207         NodeList naslovList = firstPersonElement.getElementsByTagName("naslov");
208         Element naslovElement = (Element)naslovList.item(0);
209         NodeList textNaslovList = naslovElement.getChildNodes();
210         out.write(" - "+ ((Node)textNaslovList.item(0)).getNodeValue().trim()+" - ");
211 
212         NodeList letoList = firstPersonElement.getElementsByTagName("leto");
213         Element letoElement = (Element)letoList.item(0);
214         NodeList textLetoList = letoElement.getChildNodes();
215         out.write(((Node)textLetoList.item(0)).getNodeValue().trim()+"</td>\n");
```

PROGRAMSKA KODA 2-23: Branje atributov iz .xml datoteke in pisanje v .html datoteko

Na koncu kode aplikacije še poskrbimo za lovljenje napak (programska koda 2-24).

```
369             }catch (Exception e) {
370                 System.err.println(e.getMessage());
371             }
372         }
373 
374         }catch (SAXParseException err) {
375         System.out.println ("** Parsing error" + ", line "
376              + err.getLineNumber () + ", uri " + err.getSystemId ());
377         System.out.println(" " + err.getMessage ());
378 
379         }catch (SAXException e) {
380         Exception x = e.getException ();
381         ((x == null) ? e : x).printStackTrace ();
382 
383         }catch (Throwable t) {
384         t.printStackTrace ();
385         }
```

PROGRAMSKA KODA 2-24: lovljenje napak

3 SKLEPNE UGOTOVITVE

Za temo te diplomske naloge smo si ogledali in opisali zgodovino umetnosti, novih medijev in spletnih tehnologij ter delovanje naprave Microsoft Kinect. Za nameček smo še opisali uporabniške vmesnike, se poglobili v kinetične uporabniške vmesnike in razvili aplikacijo, ki omogoča navigacijo in ogled prikazanih umetnin z uporabo gest. Podobne inštalacije se že uporabljajo na razstavah v muzejih ali festivalih in pritegnejo veliko zanimanje ljudi.

Naša aplikacija temelji na zelo preprostih tehnologijah: HTML, CSS in JavaScript. Uporabniku tako ni potrebno nameščati nobenih dodatnih programov in si stvar lahko ogleda kar v spletnem brskalniku. Zasluga za to gre knjižnici kinect.js, katera omogoča programiranje in uporabo Microsoft Kinecta za uporabo na spletu. Naša aplikacija je sicer dokaj preprosta, toda nakazuje potencial za uporabo v marsikateri drugi stroki, kot so na primer spletne trgovine, katalogi ali reklame... Za te namene se lahko obstoječo programsko kodo tudi preuredi, dopolni in optimizira.

V bližnji prihodnosti bomo verjetno zasledili porast v uporabi kinetičnih uporabniških vmesnikov, trenutno pa je tehnologija še v povojih. Kinect je prva večnamenska vhodna naprava, ki dejansko deluje in je dostopna za relativno ugodno ceno. Tudi dokumentacija za programiranje je še skopa oziroma omejena na določene programske jezike. Z razumevanjem funckij v kinect.js sem imel kar nekaj težav, saj dokumentacija ni bila napisana oziroma je bila pomankljivo napisana.

4 VIRI

[1] (2012) Art – Encyclopedia Britannica Online. Dostopno na:

http://www.britannica.com/EBchecked/topic/630806/art

[2] (2012) MySpace. Dostopno na:

http://www.myspace.com/

[3] (2012) YouTube. Dostopno na:

http://www.youtube.com/

[4] (2012) DeviantArt. Dostopno na:

http://www.deviantart.com/

[5] (2012) Depthcore. Dostopno na:

http://www.depthcore.com/

[6] (2012) A history of computer art. Dostopno na:

http://www.vam.ac.uk/content/articles/a/computer-art-history/

[7] (2012) Rare. Dostopno na:

http://www.rare.net/

[8] (2012) Prime Sense. Dostopno na:

http://www.primesense.com/

[9] (2012) Inside the race to hack Kinect. Dostopno na:

http://www.newscientist.com/article/dn19762-inside-the-race-to-hack-the-kinect.html

[10] (2012) Kinect for Windows. Dostopno na:

http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx

[11] (2012) OpenKinect. Dostopno na:

http://openkinect.org/wiki/Main_Page

[12] (2012) OpenNI. Dostopno na:

http://openni.org/

[13] (2012) Cl Nui. Dostopno na:

http://codelaboratories.com/nui

[14] (2012) Digitalt! Dostopno na:

http://www.kinecthacks.com/digitalt-interactive-3d-display/

[15] (2012) Kinectar. Dostopno na:

http://kinectar.org/

[16] (2012) Dance Dance Ribbon. Dostopno na:

http://www.kinecthacks.com/transforming-dance-into-beautiful-real-time-ribbons-via-kinect/

[17] (2012) Kinect turn. Dostopno na:

http://www.kinecthacks.com/kinect-turn-3d-pottery/

[18] (2012) KinectJS. Dostopno na:

http://kinect.childnodes.com/

[19] (2012) Impress.js. Dostopno na:

https://github.com/bartaz/impress.js/

[20] (2012) EKO GZS. Dostopno na:

http://www.semantika.si/reference/details/29

[21] (2012) Interaktivna info točka za podjetje Gen Energija. Dostopno na:

http://www.semantika.si/reference/details/28