Tallinna Ülikool
Digitehnoloogiate Instituut

Dota 2 Workshop Tools õppematerjal kohandatud mängude loomiseks

Bakalaureusetöö

Autor: Sander Leetus
Juhendaja: Jaagup Kippar

Autor: ..... „ ..... „ 2017
Juhendaja: ..... „ ..... „ 2017
Instituudi direktor: ..... „ ..... „ 2017

Tallinn 2017

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

…… …….(kuupäev) (autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Sander Leetus (sünnikuupäev: 21.04.1994)

1. Annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose Dota 2 Workshop Tools õppematerjal kohandatud mängude loomiseks, mille juhendaja on Jaagup Kippar, säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.
2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, ...

(allkiri ja kuupäev)

Sisukord

Sissejuhatus

Mängimine on tänapäeval väga populaarne, kuid mängude loomine nõuab algajalt, vähegi huvitava tulemuse saamiseks, üpris palju vaeva ning aega. Käesolevas bakalaureusetöös tutvustab autor programmi nimega *Dota 2 Workshop Tools* (edaspidi *D2WT*), mis annab huvilistele kõik vajalikud tööriistad, et luua ja avaldada teistele oma mänguväli.

Töö eesmärgiks on luua õppematerjal isikule, kes soovib luua mänguväljasid programmi *D2WT* abil. Materjalis peab olema kirjas programmi võimaluste kohta ning õpetama neid näidete abil. Materjal peab olema lugeja jaoks piisavalt hästi lahti seletatud, et oleks võimalik läbida iseseisvalt kõik peatükid ning saavutama rahuldava tulemuse.

Autor valis selle teema, kuna puudub eestikeelne õpetus programmile *D2WT* ning autori enda huvi nii programmi kui ka mängu vastu mille peal see loodud on. Kogukonna poolt on loodud foorum koos inglisekeelsete õpetustega, kus on olemas materjali nii algajatele kui ka juba edasijõudnutele keerulisemate näidetega. Autori arvates on see programm algajale alustamiseks väga hea, kuna on võimalik kasutada *D2WT* programmis olemasolevaid vahendeid ning tulemusi näeb koheselt.

Töö alguses annab autor ülevaate programmist *D2WT* ja sellest, kes selle valmistas ning miks see loodi. Lühidalt on juttu mängust, millel programm töötab ning kuidas need seotud omavahel on. Autor koostab tööriistadest harjutuste kogumi, mille abil on võimalik omandada *D2WT* olulised töövõtted. Nendeks töövõteteks on mänguvälja disainimine, *Lua* skriptiga mänguloogika loomine ning kasutajaliidese modifikatsioon. Lisaks antakse ülevaade *Hammer*-is olevatest tööriistadest, mille abil luuakse mänguväli. Programmiga on võimalik ka luua ka uusi mudeleid, materjale, osakesi ja helisid enda mängu jaoks. Samuti valmis mängu puhul on võimalik ka mängust film teha.

Eelnev kogemus mängudena on kasuks õppematerjali kasutajale - eriti mänguga, mille peale programm on ehitatud, *Dota 2*. Samuti ülevaade teiste inimeste *D2WT* programmiga loodud mänguväljadest, et oleks parem arusaam programmi võimalustest. Kasuks tuleb ka eelnev kogemus *Lua* programmeerimiskeelega, mis on kasutusel suuresti mänguloogika ülesehitamisel.

Õppematerjalis kasutusel olev *Hammer* tööriista abil on võimalik inimesel arendada disainimisoskust. Programmi arendajate poolt on loodud ka, mänguväljade loojatele, raha teenimisvõimalus.

1 *Dota 2* ja selle ajalugu

Dota 2 on üks populaarsemaid mänge üldse praegusel hetkel. Viimase 30 päeva keskmine samaaegselt mängijate arv ületab poole miljoni ning see arv on igakuiselt nii olnud viimased 2 ja pool aastat (Gray, kuupäev puudub). Praegusel hetkel omab antud mängu 99 miljonit inimest (Galyonkin, kuupäev puudub). Seega luues siia platvormile oma mänguväli, tähendab seda, et avaldades oma mänguväli, on juba olemas suur hulk publikut, kes võivad sellest huvitatud olla.

Eestis omab *Dota 2*-te umbes 160 000 inimest, kellest viimase 2 nädala jooksul aktiivselt on mänginud antud mängu ligikaudu 50 000 inimest (Galyonkin, kuupäev puudub).

1.1 Ajalugu

Warcraft III on mäng loodud *Blizzard Enternainment*-i poolt, mis tuli välja aastal 2002 ning see sisaldas *World Editor*-i, mille peal oli võimalik luua erinevaid modifikatsioone (Warcraft III: Reign of Chaos, kuupäev puudub). Nende hulka kuulus ka *Defense of the Ancients* (2003), lühidalt nimetati seda *DotA*-ks. Aastatega kogus *DotA* väga palju mängijaid ning aastal 2009 ostis *Valve Corporation* mängu õigused endale (Defense of the Ancients, kuupäev puudub).

Dota 2 (2013) on üks *Valve Corporation*-i poolt välja antud mäng, mis algselt oli loodud *Source* mängumootoril. Aastal 2015 tuli välja uuendus nimega *Reborn*, kus viidi mäng üle *Source 2* mängumootorile, millega tuli välja ka võimalus luua mängu modifikatsioone programmiga *D2WT* (Dota 2, kuupäev puudub).

D2WT tootja on *Valve Corporation*, mis asutati aastal 1996. Tegemist on USA videomängude tootjaga. Esimese mänguna anti välja *Half-Life* (1998), mis on aastatega kogunud endale väga suure fännibaasi (Valve Corporation, kuupäev puudub).

Dota 2 on üks mängudest, mida mängitakse üle maailma suurtel üritustel, kus rahalised auhinnad ülatuvad miljonitesse. Firma poolt korraldatud e-spordi auhinnafondidest tuleb välja nende edu. Töö kirjutamise ajal on e-spordi ajaloo suurim auhinnafond olnud $20 770 640,00 üritusel *The International 2016*. Antud üritusel mängitakse ainult ühte mängu, milleks on *Dota 2*. (e-Sports Earnings, kuupäev puudub)

1.2 Programm *Dota 2 Workshop Tools*

Süsteeminõuded on üpris kõrged, kuna tuleb tõmmata alla terve *Dota 2* mäng ning loomise ajal peab mäng koguaeg tagataustal käima. Mängu sulgemisel sulgub ka *Dota 2 Workshop Tools*. Programm on saadaval ainult *Windows* 7 või uuemal operatsioonisüsteemil. Optimaalseks tööks vähemalt 4 GB muutmälu. Graafikakaart, mis sobitub *DirectX 9* versiooniga. Kõvaketta mahtu vähemalt 19 GB. (Valve Corporation, 2009)

Programm *D2WT* loodi põhjusega, et anda inimestele võimalus avaldada teistele mängijatele enda loodud mänge, mis jooksevad *Source 2* mängumootori peal. Kõige populaarsematel mängudel ületab allatõmbamiste arv miljoni. Peaaegu 1700 erinevat mängu on loodud ja avaldatud. (Dota 2 workshop, kuupäev puudub)

2 *D2WT*-i õppematerjali koostamine

Peatüki esimeses punktis antakse ülevaade õppematerjali koostamise ettevalmistustest. Millele järgneb analüüs õppematerjali kohta ning lõppeb üldise kirjeldusega õppematerjalist.

2.1 Õppematerjali ettevalmistused

Õppematerjali koostamisel, vastame järgmistele küsimustele:

- **Miks on vaja luua *Dota 2 Workshop Tools*-ile eestikeelset õppematerjali?**

Peamiseks põhjuseks on eestikeelse materjali puudumine. Hakates kasutama *D2WT*-i, ei pea inimene ise midagi modelleerima, vaid saab kohe alustada mängu loomisega. Kasutajal on võimalus õppida põhilisi mänguvälja loomise töövõtteid. Programmi kasutamine on lihtne ning on võimalus kiirelt näha tulemusi, mis on motivatsiooniks kasutajale, et jätkata tööd antud programmiga.

- **Milline on õppematerjali sihtgrupp?**

Sihtgrupiks on inimesed, kellel on soov valmistada lihtsat mängu. Autor mõtleb lihtsa mängu all mängu, kus on võimalik kasutada olemasolevaid mudeleid ning detailidele ei ole suurt tähelepanu pööratud. Enamasti mängivad need inimesed ka *Dota 2*-te, kuhu mäng ka luuakse ning mänge, mis on loodud antud programmiga. Neil inimestel on võimalik ka raha teenida oma avaldatud mängu pealt.

- **Mida peab materjal sisaldama?**

Esmalt peab õppematerjalis mainitud olema, kuidas *D2WT* arvutisse paigaldada. Seejärel peab sisaldama õpetus programmi algseadistamise kohta, et oleks võimalik alustada mängu loomisega. Peale seda järgneb programmis olevate tööriistade loend ning lühike seletus iga tööriista kohta.

Õppematerjal peab kirjeldama kuidas luua mänguvälja ning selle loomise võimalustest. Seletatud peavad olema erinevad tööriistad, mida kasutada saab. Peab olema ka kirjeldatud, kust leiab valmis olevaid objekte ning kuidas neid kasutada.

Materjalis peab olema kirjas mänguloogika skriptimisest ning kuidas need mänguväljaga koos tööle panna. Kuidas objektile, mis on mänguväljal, luua skript. Lisaks peab sisaldama juttu kogukonna poolt loodud tööriistadest, mis hõlbustavad skriptide kirjutamist.

Kirjutatud peab olema ka kuidas luua uusi paneele kasutajaliidesele ning iga paneeli eraldi muuta. Lisaks peab kirjas olema kus kasutajaliidese loomise jaoks vajalikud failid on ja kuidas neid kasutada.

2.2 Õppematerjali analüüs

Tarkvara, mida kasutame on *Valve Corporation*-i poolt valmistatud *Dota 2 Workshop Tools*. Õppematerjalis kõik kasutusel olevad pildid on loodud autori poolt.

Õppematerjali kirjutamisel on autor jälginud, et vajalikud tööriistad ja mõisted oleks lahti seletatud. Paremini arusaamiseks on loodud näited ning juurde on lisatud pildid. Alguses on välja toodud lisa programmid, mis lihtsustavad töötamist *D2WT*-iga.

Õppematerjalis olevad peatükid on seatud loogilisse järjekorda, et ei oleks probleeme puudustega informatsioonis. Iga peatükk sisaldab olulist teavet, mida läheb tarvis mängu loomisel.

Isik, kes on läbinud õppematerjali on omandanud piisavalt teadmisi, et luua oma algne mänguväli. Põhjalikumate teadmiste ning olenevalt kui keerulist mänguvälja soov on luua, tuleb lisainformatsiooni juurde lugeda inglisekeelsetest õpetustest.

2.3 Õppematerjali ülesehitus

Olulisemad teadmised on lisatud õppematerjali, mis võimaldab kasutada programmi *D2WT* algtasemel.

Esimeses peatükis on õpetus kuidas soetada endale *D2WT*. Samuti on seal kirjas šablooni kohta, mis on kogukonna poolt loodud, et lihtsustada mänguvälja loomist. Kirjas on ka tööriistadest, mis on programmiga kaasas.

Teises peatükis tutvustatakse *Hammer*-it ning antakse ülevaade seal olevatest tööriistadest. Samuti on loodud näited *Tile Editor* tööriista kasutamisest.

Kolmandas peatükis luuakse mänguväli kasutades *Hammer*-it. Lisaks on juttu mis võimalused on tööriistal, et luua üleminekuid kahe erineva maa-ala vahel. Seletatud on ka kuidas lisada mänguväljale üksuseid ning nende omadusi muuta.

Neljas peatükk on suunatud skriptimisele. Kirjas on, kus asuvad vajalikud failid kuhu luua uusi skripte. Loodud on näite skript, mille funktsioonid on lahti seletatud.

Viiendas peatükis on juttu panoraamist. Antakse ülevaade panoraamist, kus asuvad selle vajalikud failid ning mis antud failid teevad. Näitena on loodud uus paneel, kuhu on tehtud mängusisesed ülesanded.

2.4 Õppematerjali testimine

Õppematerjali valmimisel andis autor materjali testijale. Testimine võttis aega umbes 2 tundi. Kõigepealt läbis testija materjali, peale mida küsis autor testijalt küsimusi. Küsimuste hulka kuulusid millega neil probleeme tekkis ja kas tulemusega jäädi rahule.

Õppematerjali testis üks isik, kes õpib informaatikat ja on varasemalt kokku puutunud mänguga *Dota 2*, kuid ei ole varem taoliste programmidega kokku puutunud. Testijal õppematerjaliga probleeme ei tekkinud, *Hammer*-iga töötamine ei olnud testija jaoks üldse raske. Testija jaoks osutus kõige raskemaks osaks *Lua*-ga skriptimine, kuna puudus varasem kokkupuude *Lua*-ga.

Esmalt alustas testija ingliskeelse kokkuvõtte lugemisega ja *barebones* šablooni alla laadimisega, peale mida hakkas õppematerjali algusest läbi töötama. Kõige enam võttis aega *Hammer*, kuna testija proovis seal erinevaid tööriistu läbi. *Lua* skripti tööle panek oli ainuke osa, mis esimese korraga tööle ei läinud, kuid programmi taaskäivitamise järel läks skript ilusti tööle.

3 Mänguväli

Programmi õppimise käigus tegi autor ise ka ühe lihtsa mänguvälja. Mänguidee on lihtne, kus automaatselt jookseb taimer ning mängijale kuvatakse selle põhjal tekst koos värviga ning mängija peab võimalikult kiiresti valima värvi mis on kirjas, mitte mis värvi kiri on. Võimalikud värvid mida valida saab on mängus all keskel reas. Kui mängija vastab valesti või liiga aeglaselt siis kaotab ta elu, kokku on 3 elu. Iga õige vastuse korral suureneb skoor ühe võrra. Igakorraga lüheneb aeg, et valida õige värv. (vt joonis 1)

Sellise idee valis autor, kuna soovis luua midagi, mis pole *Dota 2*-ga üldse seotud. Millega soovib autor näidata, et loodud mänguväli ei pea alati olema sarnane *Dota 2*-ga, mille peal neid luuakse.

Mängu kood on üleval *github*-is https://github.com/Homicida/Bakalaureusetoo. Allatõmbamisel tuleb *game* ja *content* kaustad ühendada *dota 2* kaustas olevate *game* ja *content* kaustadega. Antud *github*-is on olemas ka õppematerjali käigus loodud failid. Loodud mäng on kaustas nimega *barebones*. Õppematerjali käigus loodud failid on kaustas *barebones2*. Peale seda on *D2WT* käivitamisel *Custom Games* listis näha *barebones* mängu. Mängu saab käivitada avades konsool ning kirjutades sinna käsk *dota_launch_custom_game barebones barebones*.

Joonis 1. Autori poolt loodud mäng

Mängu loomisel on autor kasutanud kogukonna poolt loodud *barebones* šablooni. *Hammer*-iga pole autor selle mängus juures midagi teinud, kuna antud mängus on muudetud ainult mänguloogikat.

Mängu kogu *Lua* skript on *gamemode.lua* failis, muid *Lua* faile pole muudetud. Loodud on taimer funktsioonis *OnHeroInGame*, mängu alustamiseks ning ka mänguloogika jaoks vaja olev taimer funktsioonis *OnGameInProgress*, mis siis kuvab värve ning vajadusel eemaldab mängijalt elu. Loodud on funktsioon *Game*, mis kuulab vajutusi, et määrata ära kas mängija on õige värvi valinud ning saadab paneeli, et uuendada skoori. Funktsioon *ApplyCustomModifier* võtab kasutusele autori poolt loodud modifikatsiooni, mida saab kasutada, et takistada mängu, tegelaskujult elu võtmist.

Kaustas *game/dota_addons/barebones/scripts/npc* on autor muutnud faili *npc_abilities_custom.txt*, et luua antud 6 värvi valikut, mida mängus vaja läheb. Failis *npc_heroes_custom.txt* on autor loonud tegelaskuju, kellel on need 6 oskust ning vajalikud *stats*-id ning ikoonid nendeks oskusteks saab kätte kaustast *resource/flash3/images/spellicons*. Järgmine fail *npc_items_custom.txt* on loodud modifikatsioon, mis kutsutakse välja *Lua* funktsioonis *ApplyCustomModifier*.

Paneeli loomisel on autor kaustanud panoraami ning on loonud vastavalt *layout/custom_game*, *scripts/custom_game* ja *styles/custom_game* kaustadesse *scoreboard* failid. Mis annavad panoraamile välimuse ning loovad skripti, et uuendada tablood vastavalt vajadusele.

Kokkuvõte

Bakalaureusetöö eesmärgiks oli koostada ülevaade ning eestikeelne õppematerjal programmist *Dota 2 Workshop Tools*. Enne õppematerjali koostamist õppis autor selle kasutamise ära ja koostas ise ühe algse mänguvälja. Selleks uuris autor internetist programmi kohta üldist informatsiooni ja luges läbi kogukonna poolt loodud õpetusi. Seejärel koostas võimalikult kergelt arusaadava õppematerjali.

Töö käigus lõi autor veelgi ühe mänguvälja, kus loodi näited antud õppematerjali loomiseks. Mõlemad autori poolt loodud mänguväljad on ka saadavad *github*-is aadressil https://github.com/Homicida/Bakalaureusetoo. Mille allatõmbamisel saab lugeja need käivitada ning mängida.

Teema täiendamiseks on veel võimalik kirjutada teistest tööriistadest, mis on programmis olemas, et luua iseseisvalt mudeleid, osakesi ja helisid oma mängule. Samuti on võimalus ise luua oma tegelaskujusid ning anda neile vastavad omadused.

Summary

Dota 2 Workshop Tools Studying Paper for Creating Custom Games

Bachelor's thesis

This bachelor's thesis focus is on creating a studying paper for the Dota 2 Workshop Tools software. In doing so author also created his own custom game.

In preparation to create studying paper next questions will be answered:

- Why is it necessary to create estonian Dota 2 Workshop Tools studying paper?
- What is the target group of the studying paper?
- What must the studying paper contain?

In the first chapter, author writes about the game Dota 2, its popularity and history about it. First chapter also contains information about the software. Second chapter focuses on stuying paper preparation, analysis and structure. The last chapter is about the custom game author created while studying the software.

Custom games created by author can be found at https://github.com/Homicida/Bakalaureusetoo. There are 2 different games, one was made while studying the software and the other one was built while creating studying paper for examples.

Kasutatud kirjandus

Defense of the Ancients. (kuupäev puudub). *Wikipedia*. Loetud 13. aprill 2017 aadressil https://en.wikipedia.org/wiki/Defense_of_the_Ancients

Dota 2. (kuupäev puudub). *Wikipedia*. Loetud 13. aprill 2017 aadressil https://en.wikipedia.org/wiki/Dota_2

e-Sports Earnings. (kuupäev puudub). *Tournaments*. Loetud 13. aprill 2017 aadressil http://www.esportsearnings.com/tournaments

Galyonkin S. (kuupäev puudub). Steam spy. Loetud 26. aprill 2017 aadressil http://steamspy.com

Gray J. (kuupäev puudub). Steam Charts. Loetud 26. aprill 2017 aadressil http://steamcharts.com/app/570

Source (game engine). (kuupäev puudub). *Wikipedia*. Loetud 13. aprill 2017 aadressil https://en.wikipedia.org/wiki/Source_(game_engine)

Valve Corporation. (kuupäev puudub). Dota 2 workshop. Loetud 13. aprill 2017 aadressil http://steamcommunity.com/app/570/workshop/

Valve Corporation. (kuupäev puudub). Valve. Loetud 13. aprill 2017 aadressil http://www.valvesoftware.com/

Valve Corporation. (2009). Dota 2. Loetud 13. aprill 2017 aadressil http://store.steampowered.com/app/570/

Warcraft III: Reign of Chaos. (kuupäev puudub). *Wikipedia.* Loetud 13. aprill 2017 aadressil https://en.wikipedia.org/wiki/Warcraft_III:_Reign_of_Chaos

Tallinna Ülikool
Digitehnoloogiate Instituut

Dota 2 Workshop Tools õppematerjal kohandatud mängude loomiseks

Bakalaureusetöö

Autor: Sander Leetus
Juhendaja: Jaagup Kippar

Autor: ..... „ ..... „ 2017
Juhendaja: ..... „ ..... „ 2017
Instituudi direktor: ..... „ ..... „ 2017

Tallinn 2017

Sisukord

Sissejuhatus

Esimeses peatükis on seletatud kuidas paigaldada programm *D2WT* arvutisse. Järgnevas peatükis on loodud ülevaade erinevatest tööriistadest, mida saab *Hammer*-is kasutada. Sellele järgneb peatükk, kus luuakse mänguväli kasutades *Hammer*-it. Mänguväljale luuakse algne välimus, koos mõne vahendiga, et näidata kuidas nende omadusi on võimalik muuta.

Neljandas peatükis luuakse loodud mänguväljale skript, mis kasutab vahendeid, mis sai lisatud eelnevalt *Hammer*-is. Õppematerjali viimases peatükis luuakse kasutajaliidesele uus paneel.

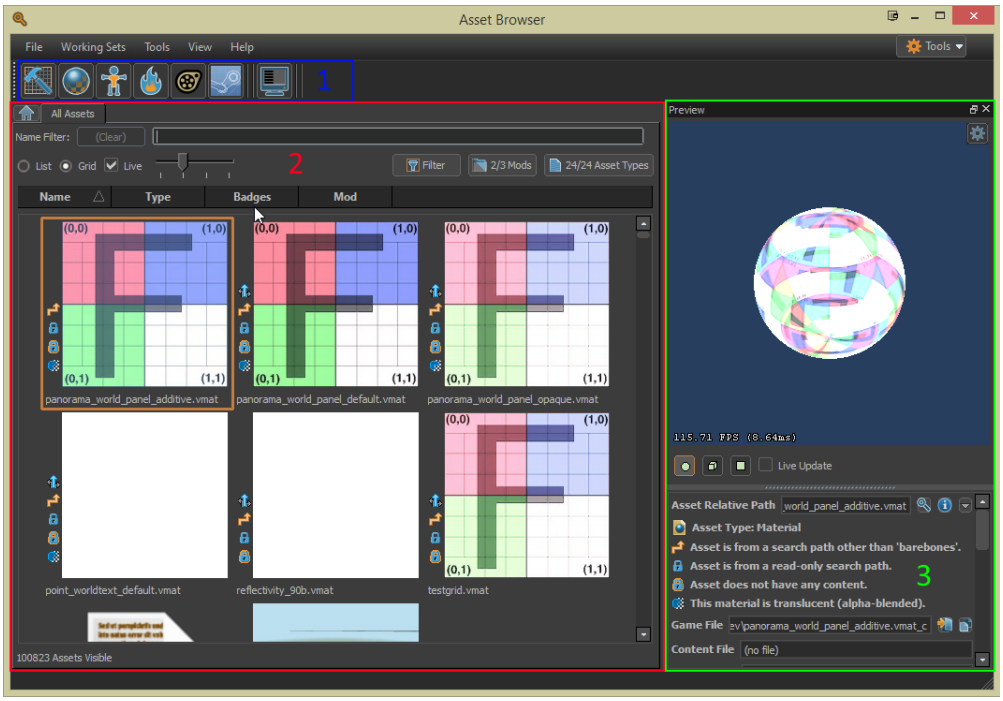
Kõik näidetes kasutusel olevad vahendid on programmis eelnevalt olemas. Uute materjalide ja vahendite loomist antud õppematerjal ei käsitle.

1 Programmi *Dota 2 Workshop Tools* paigaldamine arvutisse

Esimese sammuna tuleb arvutisse paigaldada programm nimega *Steam*. Lingi allalaadimiseks leiab veebilehelt http://store.steampowered.com/ ülevalt paremalt nurgast. Peale *Steam*-i paigaldamist tuleb sinna ka kasutaja teha. Kui oled sisse loginud, saad *Store* nupu alt programme otsida. Otsingusse kirjutada *Dota 2* tuleb ette vastav mäng ning see avades vajutada nupule *Install*. Mängu suurust arvestades võtab allatõmbamine aega ning kui pakutaval interneti lahendusel on limiteeritud andmemaht, siis peaks ka seda jälgima.

Kui *Dota 2* on paigaldatud, siis leiab selle *Library* peale vajutades. Peale seda vajuta paremklõps *Dota 2* peale ja vali *View Downloadable Content*, mille järel tee linnuke *Dota 2 Workshop Tools DLC Install veergu*. Vajuta *Close* ning *Steam* hakkab vajalikku sisu tõmbama.

Asjade lihtsustamiseks on kogukonna poolt tehtud *Barebones* šabloon, mille leiab https://github.com/bmddota/barebones lehelt. Peale allalaadimist, mine *...\Steam\steamapps\common\dota 2* ning ühenda *content* ja *game* kaustad allatõmmatud zip arhiivist *dota 2* kausta. Järgmisena käivita *Dota 2*, *m*ängu tööle pannes tuleb ette valik, kust tuleb valida *Launch Dota 2 – Tools*. Avaneb aken kus all osas on *Custom Games* ning seal listis on ka nüüd uus *Addon* nimega *barebones*, misjärel avaneb *Asset Browser* kui teha sellepeale topeltklõps (vt joonis 2).

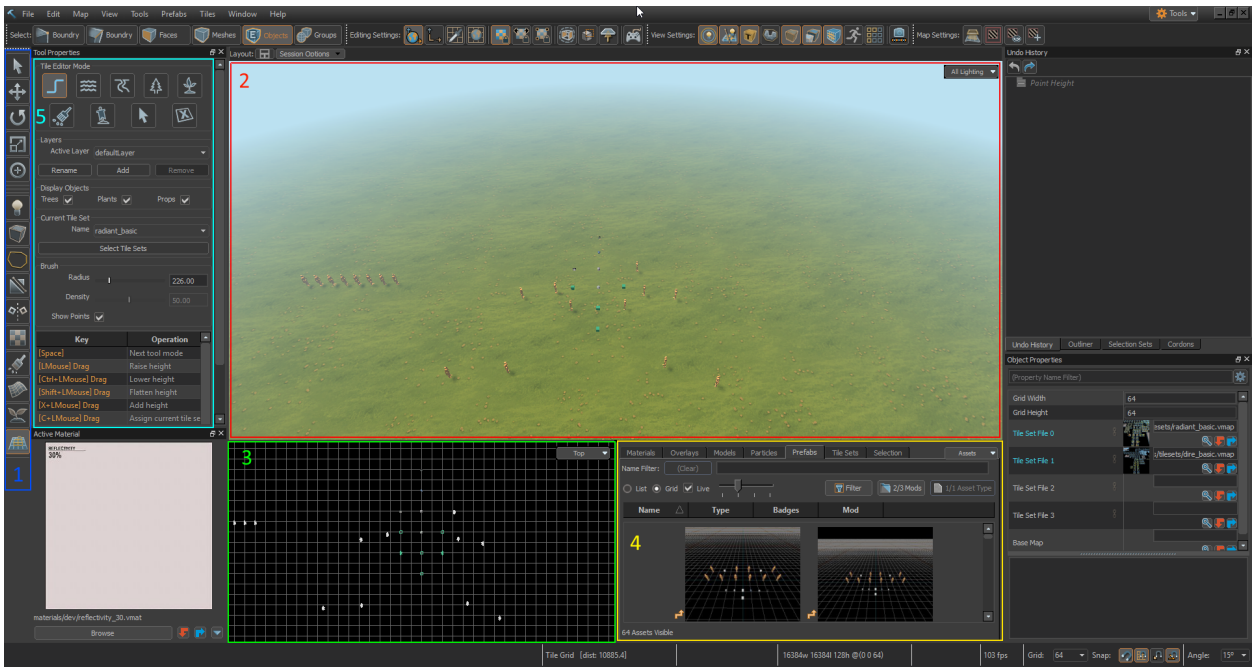


Joonis 2. *Asset Browser* ülesehitus

1. *Tools* - Erinevad tööriistad mida on võimalik kasutada. Vasakult paremale loetelu:
    - *Hammer (Map Editor)* - Tööriist, millega saab luua mänguväljasid *Dota 2* jaoks.
    - *Material Editor* - Seda kasutatakse, kui on soov luua uusi või muuta olemasolevaid materjale oma loodud mängu jaoks.
    - *Model Editor* - Olemas on ka tööriist, millega on võimalik luua uusi mudeleid.
    - *Particle Editor* – Osakeste muutmiseks ning loomiseks on samuti olemas tööriist.
    - *Source Filmmaker* - Seda saab kasutada, et teha mängude kohta videosid.
    - *Workshop Manager* - Selle abil saab teistele inimestele avalikustada enda loodud mängu.
    - *Console* - Avab konsooli
2. *Assets* - Näitab ära kõik võimalikud vahendid, mida on võimalik kasutada ja avada vastada tööriistaga. Kasutada on võimalik ka filtreid, et kergemini üles leida vahend mida kasutada soovid.
3. *Preview* - Näitab eelvaadet, valitud vahendist. Sisaldab ka informatsiooni vahendi kohta.

2 Hammer

Peale *Hammer*-i avamist, minna *file -> open* ja avada sealt *template_map.vmap*. Avaneb mänguväli, kuhu on lisatud juba ruudustik ja mõned üksused, et oleks võimalik mänguvälja tööle panna ja testida (vt joonis 3).



Joonis 3.Hammeril olevad alad

1. Tööriistad
    - *Selection Tool* – Tööriist on vaikimisi valitud *Hammer*-i avamisel. Kõige tavalisem viis, et valida geomeetrilisi elemente mänguväljal. Peale valiku tegemist on võimalik elemente manipuleerida teiste tööriistadega. Kiirklahv tööriista valimiseks on *Shift+S*.
    - *Translate Tool* – Võimaldab liigutada valitud elemente mööda X-, Y- või Z telgjoont. Mööda X-telgjoont liigustamiseks on punane nool, roheline nool Y-telgjooneks või sinist nöölt Z-telgjooneks. Liigutada on võimalik ainult mööda ruudustikku, kuid hoides all *Ctrl* nuppu, võimaldab see vabalt liigutada. Kiirklahv tööriista valimiseks on T.

- *Rotate Tool* – Tööriist, mille abil on võimalik pöörata elemente piki telgjoont. Liigutamiseks kasutatakse punast, rohelist ja sininst ringi. Mille lohistamisel liigub element piki vastavat telgjoont. Nurk, mille all pöörlemine toimub on määratud, all paremas nurgas, kuid hoides all nuppu *Ctrl* pööramisega hetkel eirab see nurga seadet. Kiirklahv tööriista valimiseks on R.
- *Scale Tool* – Selle tööriista abil on võimalik muuta elementide suurust. Suuruse muutmisel tuleb lohistada kuubikuid. Sarnaselt eelnevatele tööriistadele, on ka sellel tööriistal punane, roheline ja sinine vastavalt X-, Y- ja Z-telgjoon. Võimalik on ka igas suunas suurust muuta, kasutades selleks lillat kuubikut. Kiirklahv tööriista valimiseks on E.
- *Pivot Tool* – Võimaldab muuta valitud elementide pöördepunkti. Näiteks kuubi korral on pöördepunkt keskel ning kuup pöörleb kohapeal, kuid pöördepunkti muutes on võimalik kuupi pöörata ümber ühe nurga. *Insert* klahviga saab avada ja kinni panna ning *End* klahv lähtestab pöördepunkti.
- *Entity Tool* – Võimaldab lisada mänguväljale üksuseid, nende hulka kuuluvad valgus, tegelaskujude alguspunktid, rekvisiidid ja palju muud. Kiirklahv on *Shift + E*.
- *Block Tool* – Saab luua tavalisi geomeetrilisi kujundeid. Võimalik on luua nelinurka, risttahukat, silindrit, koonust ning kera. Kiirklahv on *Shift + B*.
- *Polygon Tool* – Võimaldab luua hulknurkset kujundit.
- *Clipping Tool* – Selle tööriistaga on võimalik lõigata elemente kolmel erineval viisil. Olenevalt viisist jääb element alles ühelt poolt lõiget, teiselt poolt või mõlemad uued elemendid jäävad alles. Tööriista peab kasutama 2D vaates ning peab kinnitama vajutades *Enter*, muidu tühistatakse muudatused. Erinevaid viise saab vajutades kiirklahvile *Shift + X*.
- *Mirror Tool* – Võimaldab peegeldada valitud elemente, vastavalt kasutaja valitud telgjoont mööda. Kiirklahv on *Shift + F*.
- *Texture Projection Tool* – Tööriistaga on võimalik teha tekstuuri vastendamist, mis on kasulik kui on vaja täpsustada keerulisi geomeetrilisi kujundeid. Kiirklahv on *Shift+P*.

- *Paint Tool* - Antud tööriistaga on võimalik materjalidel omavahel seguneda. Vajaduse korral on võimalik ka kasutada antud tööriista, et luua üleminekuid ühelt alalt teisele. Kiirklahv on *Shift + V*.
- *Asset Spray Tool* – Pooleli olev tööriist, mis võimaldab kõiki vahendeid värvida mänguväljale.
- *Tile Editor* – Mänguvälja loomisel enim kasutatud tööriist. Võimaldab üksteist kokku ühendada eelmääratud plaatidel, mis moodustavad mängitava mänguvälja. Juhul kui mänguväljal puuduvad plaadid, siis genereeritakse automaatselt suur ala, kuhu saab mänguvälja luua.

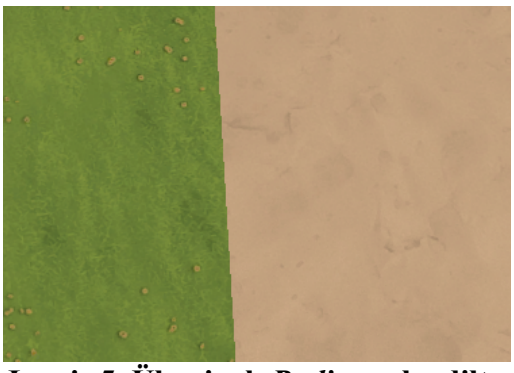
2. 3D vaade – Näitab 3D ülevaadet mänguväljast ja laseb sul navigeerida ringi. Suurem osa tööst toimub siin vaates. Mänguvälja edasi liigutamiseks on klahv W, vasakule A, paremale D, tagasi S ja pööramiseks paremklõpsu. Vajutades klahvi Z aktiveerub lendamine ilma, et peaks hiirenuppu all hoidma, WASD-i kasutatakse endiselt. et liikumist kontrollida.
3. 2D vaade – Annab sulle ülalt, alalt ja kõrvalt vaate mänguväljast. Kiirklahvid vaadete muutmiseks: ülalt F2, eest F3 ja kõrvalt F4.
4. *Assets* - Sarnane ülesehitus *Asset Browser*-iga, mis sisaldab sisu loomisel kasutuseks minevaid vahendeid.
5. *Tool Properties* - Tööriista omadused, välja toodud pildil on valitud *Tile Editor* tööriist, mida läheb kõige rohkem mänguvälja loomisel tarvis.

3 Mänguvälja loomine

Mängu loomisega alustamiseks, tuleb kõigepealt luua mänguväli. Esmalt alustasin *Paint Terrain* tööriistaga, millega lõin ringikujuline maaala ning seejärel ümbritsesin loodud ala veega ja jagasin neljaks. Peale seda muutsin ära kolmel alal maa-ala, milleks valisin *Radiant snow*, *Radiant desert* ja *Dire basic*. Igale alale lisasin keskele tee, et tekiksid trepid ja oleks võimalik sinna liikuda. Seejärel lisasin mõned puud ja põõsad (vt joonis 4).

Joonis 4. Loodud mänguväli

Kasutasin vett alade jaotamiseks, kuna hetkel on võimalik ainult *Dire* ja *Radiant* välimusega normaalset üleminukut teha. Teiste välimustega tuleb loovaks hakata (vt joonis 5).



Joonis 5. Üleminek *Radiant* plaadilt *Radiant Desert* plaadile

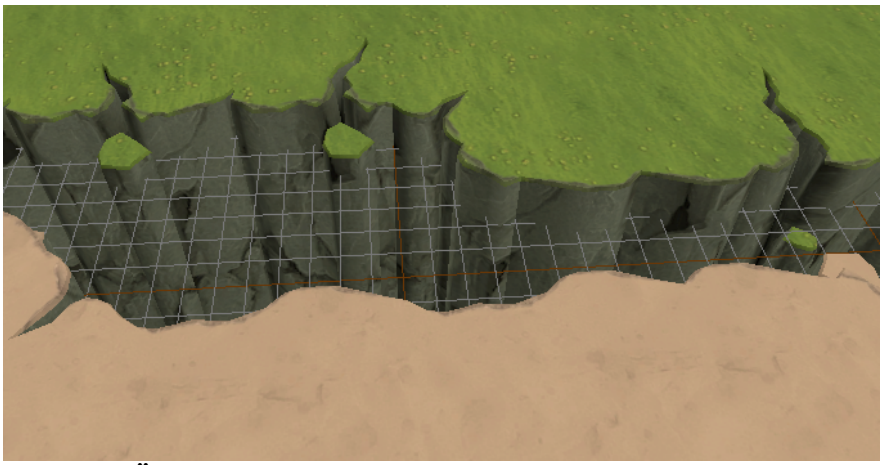
Kuna *Paint Blends* tööriistas pole midagi mis võimaldaks, luua üleminekut antud kahe plaadi vahel, siis tuleb loovaks hakata. Antud näites olen selleks kasutanud *Paint Blend*-is *Layer* 2-te, et luua mõlemale alale kivist teerada ning seejärel valisin *Layer* 1-e, millega peitsin kivist teerada ning sellega muutsin värvid natukene paremini sulanduvaks. (vt joonis 6)

Joonis 6. Ülemineku loomine *Paint Blends* tööriistaga

Seejärel, et peita ära ka plaatide vahel olev triip, kasutasin selleks *Place Objects* tööriista ning lisasin triibule peale rekviisiite nimega *path_overlays*, mille tulemus ei ole võibolla just kõige parem, kuid kuna hetkel antud tööriist on limiteeritud sellel alal siis tuleb töötada tööriistadega, mis saadaval on (vt joonis 7).

Joonis 7. Valmis üleminek

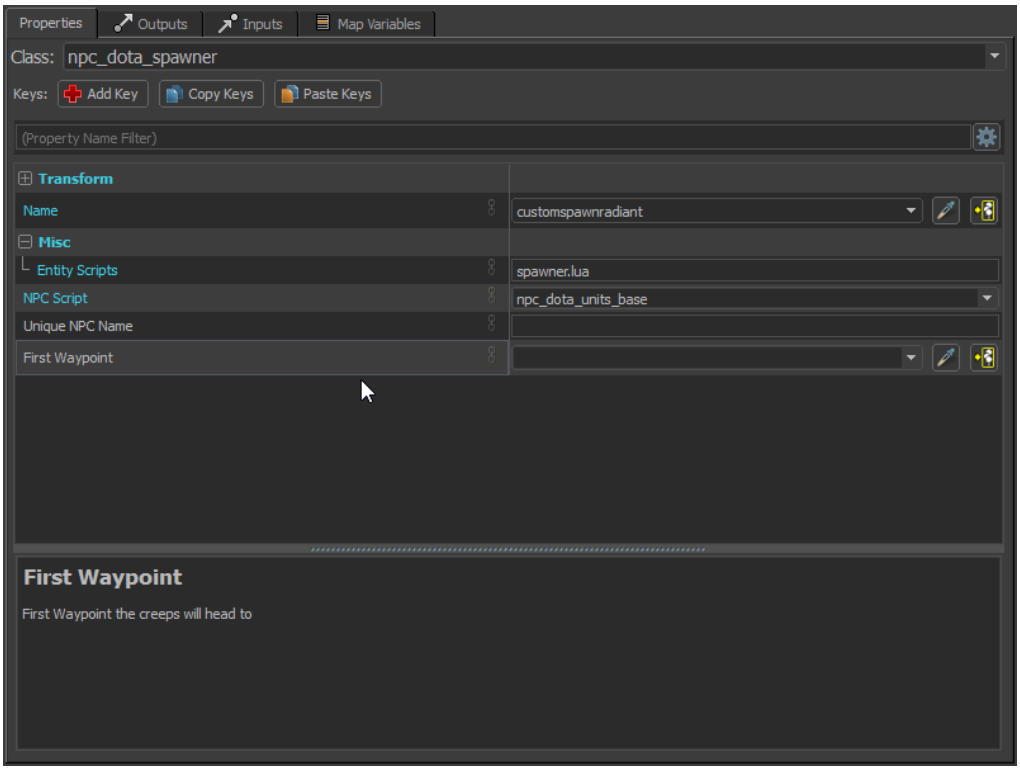
Võimalik on ka, lihtsalt maa-ala nende vahel ära hävitada, kuid seljuhul pole võimalik seal liikuda. See võimalus on *Paint Path* tööriista all, kui *Path Type* panna *destruction*. Tulemus näeks välja siis selline (vt joonis 8).



Joonis 8. Ülemineku loomine kasutades *destruction* võimalust

Kui mänguvälja välimus on valmis, siis on aeg lisada mänguloogika jaoks vajaolevaid vahendeid. Antud näites olen võtnud *Prefabs*-i all poe *shop_keeper_radiant* ja lisanud selle kõige keskele. Vajutades *Alt + Enter* avaneb objekti omadused eraldi aknas ning on kergem neid muuta. Poe lisamisel veenduda, et omaduste all tüüp oleks *Home*. Peale seda valisin *Entity* klassi

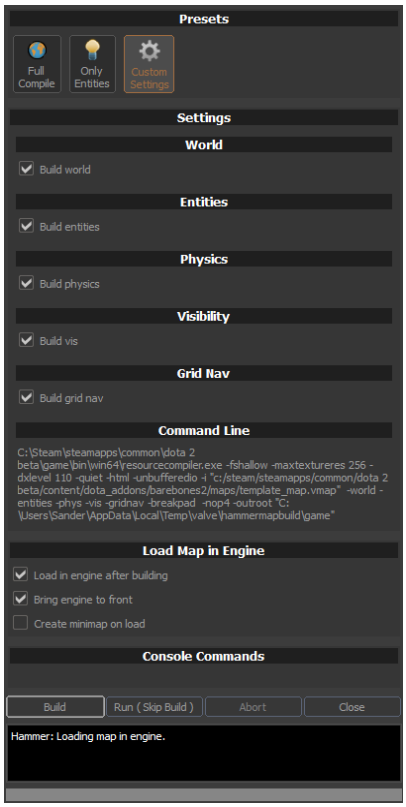
nimega *npc_dota_spawner* ja lisasin ühe *Radiant* alale. Seejärel muutsin vastavalt pildile objekti omadused (vt joonis 9).



Joonis 9. Objekti omaduste muutmisaken

Lisasin ka *Dire* alale sama üksuse, ainult nime muutsin ära, seal kasutasin nime *customspawndire*.

Vajutades *F9*, avan mänguvälja ehitamise ja käivitamise akna. Esmasel ehitamisel tuleb linnuke teha ka *Create minimap on load* ette kasti. Seejärel vajutada nuppu *Build*. Kui mänguvälja ennast ei muudeta vaid ainult tegeletakse skriptimisega, siis võib ka lihtsalt mäng käivitada, jättes vahele ehitamise osa (vt joonis 10).



Joonis 10. Seaded, et ehitada ning käivitada mänguväli

Seejärel, käivitub mäng ning kuvatakse ette tiimide list. Antud näites vahet pole millises tiimis olla, seega seda ei hakka muutma ning võib vajutada nuppu *LOCK + START*. Peale seda kuvatakse ette valik kõikide tegelaskujudega, mis on saadaval mängus. Antud olukorras ei muuda tegelaskuju mitte midagi, seega võib valida ühe *RANDOM* nupu abil ning *SKIP AHEAD* nuppu vajutades saab vahele jätta ootamisaja, mis on määratud mängu poolt.

4 Skriptimine LUAga

Skriptimine *Dota 2*-s käideldakse läbi virtuaalmasina *Vscript*, mis töötab kui abstraktne sidekiht mängumootori ja väliste skriptide vahel. Kasutatakse *Lua* programmeerimiskeelt. Skriptidega saab kontrollida sündmusi, mis toimuvad mängu režiimides, mängu reeglites, tegelaskuju oskustes ja koostoimes, neutraalsed vastased, tehisintellekt ja palju muud.

Lua skript failid asuvad *.../game/dota_addons/barebones/scripts/vscripts* kasutas. Kuna olen kasutusele võtnud *Barebones* šablooni, siis leiab sealt erinevaid näiteid, mängu reegleid, mida saab kergelt muuta ja teegid.

Kui *spawner*-id on mänguväljale lisatud, tuleb neile skript kirjutada. Selleks loon uue faili nimega *spawner.lua vscripts* kausta (vt koodinäide 1).

```
SPAWNLOCATIONDIRE = "customspawndire"
SPAWNLOCATIONRADIANT = "customspawnradiant"

if customSpawn == nil then
  customSpawn = class({})
end

function Precache(context)
  PrecacheUnitByNameSync("npc_dota_creep_badguys_melee",context)
  PrecacheModel("npc_dota_creep_badguys_melee", context)
  PrecacheUnitByNameSync("npc_dota_neutral_kobold", context)
  PrecacheModel("npc_dota_neutral_kobold", context)
end

function customSpawn:spawnunits()
  local spawnLocationDire = Entities:FindByName(nil,
SPAWNLOCATIONDIRE)
  local enemy = CreateUnitByName("npc_dota_creep_badguys_melee",
spawnLocationDire:GetAbsOrigin(),true,nil, nil,
DOTA_TEAM_BADGUYS)
  local i = 0
  local spawnLocationRadiant = Entities:FindByName(nil,
SPAWNLOCATIONRADIANT)
  while i < 5 do
    local enemy2 = CreateUnitByName("npc_dota_neutral_kobold",
spawnLocationRadiant:GetAbsOrigin(), true, nil, nil,
DOTA_TEAM_BADGUYS)
    i = i +1
  end
end
```

```
function Activate()
  GameRules.customSpawn = customSpawn()
  GameRules.customSpawn:InitGameMode()
end

function customSpawn:InitGameMode()
  GameRules:GetGameModeEntity():SetThink("spawnunits", self)
end
```

Koodinäide 1. *spawner.lua* faili sisu

Funktsioonis *Precache* laen mängu vastavad mudelid mida kasutama hakkan. *spawnunits* funktsioonis määran ära asukoha üksuste järgi, mille lisasin eelnevalt mänguväljale ning seejärel loon antud asukohale vastased. Funktsioonide *Activate* ja *InitGameMode* loon mängureeglitesse vastavad uued klassid ja jooksutan funktsioonid.

Peale faili loomist ja salvestamist taaskäivitan mängu, kas *Hammer*-ist mänguvälja ehitamisega või kasutades konsooli käsku *dota_launch_custom_game barebones template_map*. Mängu käivitamisel on näha uued vastased (vt jooniseid 11 ja 12).

Joonis 11. Skripti abil loodud vastane

Joonis 12. Skripti abil loodud vastased

5 Panoraam

Panoraam on *Valve* loodud kasutajaliidese raamistik, mis on tugevalt mõjutatud ning meenutab tänapäeva veebilehele koostamist. See võimaldab kiiret arendamist, kõrget kvaliteeti ja jõudlust.

Kui skriptimine toimus *.../game/dota_addons/barebones/scripts/vscripts* kaustas, siis tuleb minna tagasi *content* kausta kus asusid ka mänguväljad *.../content/dota_addons/barebones/panorama/*.

5.1 Ülevaade panoraamist

Panoraam koosneb paneelidest, veebilehe terminis paneel on *HTML* element. Kõik mida sa kasutajaliideses näed – pildid, nupud ja sildid - on paneelid.

Andmete skelett defineeritakse ära XML failides, seal kirjeldatakse ära millised paneelid kuvatakse kasutajaliideses ning nende hierarhiline ülesehitus. XML failid leiab *layout/custom_games/* kaustast.

Kasutajaliidese välimuse saab ära määrata CSS failides. Konsooli käsu *dump_panorama_css_properties* saab kätte kõige uuema dokumentatsiooni. Samuti wiki lehel https://developer.valvesoftware.com/wiki/Dota_2_Workshop_Tools/Panorama/CSS_Properties on kogu dokumentatsioon välja kirjutatud. CSS failid leiab *styles* kaustast.

Panoraamides kasutatakse *Javascript*-i skriptimiseks, mis võimaldab kasutajaliidesel reageerida vastavalt kasutaja sisenditele ja mängu sündmustele. *Javascript* kood on ka võimeline suhtlema mängu serveri poolse koodiga, mis on rakendatud *Lua*-s. *Javascript* failid asuvad *scripts* kaustas.

Event-i abil on tehtud võimalikuks paneelide vaheline suhtlus. Sündmuste abil on võimalik lihtsustada paljusid levinuid ülesandeid, kasutajaliidese loomisel. Wiki lehelt https://developer.valvesoftware.com/wiki/Dota_2_Workshop_Tools/Panorama/Events leiab kõik omadused ning sündmused, mida panoraam toetab.

5.2 Paneeli loomine mängule

Järgnevalt loon enda mängule paneeli koos ülesannetega. Selleks alustan uue XML faili loomisega. Loon faili nimega *quest.xml layout/custom_game* kausta. Selleks, et mäng üldse

antud faili kasutusele oskaks võtta, pean samas kaustas oleva *custom_ui_manifest.xml* failile ühe rea juurde lisama (vt koodinäide 2).

```
<CustomUIElement type="Hud" layoutfile="file://
{resources}/layout/custom_game/quest.xml" />
```

Koodinäide 2. Loeme sisse uue XML faili

Peale seda avan *quest.xml* faili ja loon algse koodi puu. Samuti lisan ja loon juurde ka CSS ja JS failid vastavalt *styles* ja *scripts* kaustadesse (vt koodinäide 3).

```
<root>
  <styles>
    <include src="file://{resources}/styles/quest.css" />
  </styles>
  <scripts>
    <include src="file://{resources}/scripts/quest.js" />
  </scripts>
  <snippets>
  </snippets>
  <Panel>
  </Panel>
</root>
```

Koodinäide 3. XML faili algne koodi puu

Snippets-is loon jupi, mis on šablooniks ülesannetele mida saab *Panel* tüvele lisada. Loon kõigepealt `<Panel>` tüvesse paneeli (vt koodinäide 4).

```
<Panel class="QuestRoot">
  <Label id="QuestName" text="Ülesanded" />
  <Panel id="Quests">
  </Panel>
</Panel>
```

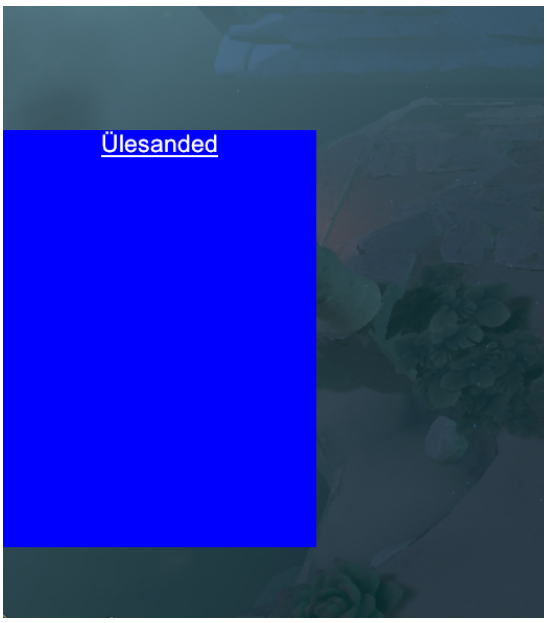
Koodinäide 4. Uus paneel

Esmasel XML faili loomisel, et tulemust mängus näha, tuleb mäng taasavada. Kuna CSS faili pole muudetud, siis tekib mängu vasakusse ülesse nurka lihtsalt tekst ülesanded. Seega järgmiseks muudan CSS faili ning annan enda paneelile asukoha (vt koodinäide 5).

```
.QuestRoot{
     width: 300px;
     height: 400px;
     horizontal-align: left;
     vertical-align: center;
     background-color: blue;
}
```

```
#Questname{
     font-size: 24px;
     color: white;
     text-decoration: underline;
     horizontal-align: center;
}
```
Koodinäide 5. Paneeli asukoht

Mille tulemusel tekib vasakule paneel koos antud stiiliga (vt joonis 13).



Joonis 13. Paneeli algvälimus

Järgmiseks loon `<snippets>` tüvele uue jupi, kus määran ära ülesande ülesehituse, mis kuvatakse ühekaupa üksteise alla. Vasakule lisan ülesannetele pildi ning paremale ülesande pealkirja, kirjelduse ja progressi (vt koodinäide 6).

```
<snippet name="Quest">
  <Panel class="Quest">
     <Panel id="LeftSide">
       <Panel id="QuestImage">
          <Image src="file://{resources}/images/quest.png" />
       </Panel>
     </Panel>
     <Panel id="RightSide">
       <Panel id="TopSide">
          <Label id="QuestTitle" text="Sample Title" />
       </Panel>
```

```
        <Panel id="BottomSide">
           <Panel id="Background" />
           <Label id="QuestDescription" text="Sample Text"/>
           <Label id="QuestProgress" text="0/1" />
        </Panel>
      </Panel>
  </Panel>
</snippet>
```

Koodinäide 6. Ülesande paneeli ülesehitus.

Antud jupis olen kasutanud ka pilti, mille laadisin alla lehelt http://i.imgur.com/e54mzNp.png ning tegin selle 50x50px suuruseks. Pildi salvestasin kausta *.../content/dota_addons/barebones/panorama/images/* nimega quest.png. Loodud juppi ei ole veel mängus näha, kuna see on kõigest šabloon ning seda tuleb alles *Javascript*-iga välja kutsuda. Seega järgmiseks avan *quest.js* faili ning kasutan šablooni, et luua ülesanne (vt koodinäide 7). Samuti kuna pole välimust loonud antud jupile, siis lisan ka selle (vt koodinäide 8).

```
function InitQuest(){
  var panel = $.CreatePanel("Panel", $("#Quests"), "");
  panel.BLoadLayoutSnippet("Quest");
}

InitQuest();
```

Koodinäide 7. Paneeli loomine *Javascript*-iga

```
#Quests{
     flow-children: down;
     margin-top: 120px;
}
.Quest{
     flow-children: right;
}
#LeftSide{
     width: 50px;
     height: 50px;
     margin: 5px 5px 5px 5px;
}
#RightSide{
     flow-children: down;
}
#BottomSide{
     width: 200px;
     margin-top: 10px;
     border: 2px solid black;
}
```

```
#QuestTitle{
	color: black;
	font-size: 16px;
	margin-top: 5px;
	margin-left: 2px;
}
#QuestProgress{
	color: black;
	font-size: 14px;
	horizontal-align: right;
}
#QuestDescription{
	color: black;
	font-size: 14px;
}
#Background{
	width: 100%;
	height: 100%;
	background-color: red;
}
```

Koodinäide 8. *Snippet*-is olevate paneelide välimus

Peale failide lisamist näeb loodud paneel välja selline (vt joonis 14).

Joonis 14. *Snippet*-ist loodud paneel, koos välimusega

Järgmiseks loon ülesande, millele annan nime ning loon funktsiooni millega saab progressi rida muuta (vt koodinäide 9).

```
function InitQuest(name, desc, target){
  var panel = $.CreatePanel("Panel", $("#Quests"), "");
  panel.BLoadLayoutSnippet("Quest");
  panel.FindChildTraverse("QuestTitle").text = name;
  panel.FindChildTraverse("QuestDescription").text = desc;
  panel.name = name;
  panel.desc = desc;
  SetQuestProgress(panel, 4, target);
  return panel;
}

function SetQuestProgress(quest, current, goal){
  var percent = (current / goal);
  quest.FindChildTraverse("QuestProgress").text = current + "/"
+ goal;
  var background = quest.FindChildTraverse("Background");
  background.style.width = (percent * 100) + "%";
  quest.goal = goal;
  quest.current = current;
}
InitQuest("Esimene ülesanne", "Ülesande kirjeldus", "15");
```

Koodinäide 9. Ülesande progressi muutmise funktsioon

Samuti kustutasin CSS failis .*QuestRoot* klassist *background-color* omaduse ära ning nüüd näeb ülesannete paneel selline (vt joonis 15).

Joonis 15. Ülesandele progressi lisamine

InitQuest funktsioon loob uue ülesande, mille pealkiri on Esimene ülesanne ja kirjeldus Ülesande kirjeldus. *SetQuestProgress* määrab ära, et kui ülesande eesmärk antud juhul on 15, siis sellest 4 on tehtud.

Loon juurde paar funktsiooni, et teha *Javascript* vastuvõtlikuks ka *Lua*-le (vt koodinäide 10). Esimesena lisan koodi algusesse rea `var quests = {};` millega loon uue massiivi nimega *quests*. `SetQuestProgress(panel, 4, target);` antud rida määrab ära ülesande progressi, aga algväärtustamisel on ülesande progress 0, seega muudan real number nelja nulliks.

```
function RemoveQuest(quest){
  quest.DeleteAsync(0);
}
function OnNewQuest(dat){
  var quest = InitQuest(dat.name, dat.desc, dat.max);
  quest.tag = dat.id;
  quests[dat.id] = quest;
}
function OnQuestUpdateProgress(dat){
  for(var x in quests){
    quest = quests[x];
    if(quest.tag == dat.id){
      SetQuestProgress(quest, dat.current, dat.max);
      break;
    }
  }
}
function OnQuestRemove(dat){
  for(var x in quests){
    quest = quests[x];
    if(quest.tag == dat.id){
      RemoveQuest(quest);
      break;
    }
  }
}
function debug(){
  GameEvents.Subscribe("quests_create_quest", OnNewQuest);
  GameEvents.Subscribe("quests_update_quest",
OnQuestUpdateProgress);
  GameEvents.Subscribe("quests_remove_quest", OnQuestRemove);
}
debug();
```

Koodinäide 10. *Javascript*-i lisatud uued funktsioonid

RemoveQuest() funktsiooni abil kustutan vajadusel ülesande. *OnNewQuest()* funktsioonis ma mitte ainult ei loo uut ülesannet, vaid lisan ta ka *quests* massiivi, mille abil saan jälgida mitut erinevat ülesannet. *OnQuestUpdateProgress()* ja *OnQuestRemove()* funktsioonide abil vastavalt kas uuendan või eemaldan vajaliku ülesande, kindla ülesande määramisel, kasutan selleks massiivi abi. Funktsioonis *debug()* kasutan *GameEvents.Subscribe()* funktsiooni, mille abil suhtlen *Lua*-ga.

Lua koodi kirjutamiseks tuleb avada kaust *.../game/dota_addons/barebones/scripts/vscripts* ja sealt hakkan *gamemode.lua* faili muutma. Siin failis on algväärtustatud kõik vajalikud funktsioonid, et mäng tööle läheks alguses. Palju on ka kommentaare pandud, mis peaks aitama algajal aru saada rohkem. See kõik on *barebones* šabloonis. Kõigepealt tuleb luua uued ülesanded ja need loon funktsioonis *Gamemode:OnGameInProgress()* (vt koodinäide 11).

Eelnevalt sai loodud mänguväljale vastased, nüüd teen vastava ülesande, nende hävitamiseks. Loon ülesande nimega *Kobolds*, kirjeldusega *Kill kobolds,* eesmärgiks määran 5 ja *id* on 1.

```
function GameMode:OnGameInProgress()

CustomGameEventManager:Send_ServerToAllClients("quests_create_qu
est"), {name = "Kobolds", desc = "Kill kobolds", max = 5, id=1})
end
```

Koodinäide 11. *Lua*-s muudetud funktsiooni sisu.

Faili alguses loon globaalse muutuja `UNITSKILLED = 0` ning lõppu teen uue funktsiooni, mis kuulab mängu sündmusi, kui mingi vastane hävitatakse ja saadab need *Javascript*-i, mis seejärel uuendab ülesannet (vt koodinäide 12).

```
function GameMode:OnEntityKilled(event)
  local killedUnit = EntIndexToHScript(event.entindex_killed)
  if killedUnit and string.find(killedUnit:GetUnitName(),
"kobold") then
    UNITSKILLED = UNITSKILLED + 1

CustomGameEventManager:Send_ServerToAllClients("quests_update_qu
est", {max = 5, current = UNITSKILLED, id = 1})
    if UNITSKILLED == 5 then

CustomGameEventManager:Send_ServerToAllClients("quests_remove_qu
est", {id=1})
    end
  end
end
```

Koodinäide 12. Funktsioon *OnEntityKilled*

Kuna *Lua*-t on muudetud tuleb mänguväli, *Hammer*-ist või konsooli käsuga, uuesti käima panna. Kui koodis vigu pole peaks tulemus välja nägema nagu pildil (vt joonis 16).

Joonis 16. Valmis paneel