30.1.2011

Poročilo za projektno delo

Predmet: Računalniška grafika

Toše Dimkov, Črtomir Drofenik in Žiga Špindler

Kazalo vsebine

Uvod

Namen dela pri tem predmetu je bil, nadgradnja lanskega projekta z grafiko in ga dopolniti, da bo nastala celotna igra. Zaradi novih zahtev za vključitev grafike in drugačnih želja za končni izdelek, smo se odločili, da zavržemo večino lanskega projekta in letošnjega začnemo od začetka.

Igra je obdržala ime Minotavrovo maščevanje in prav tako smo obdržali programski jezik C# in XNA kot ogrodje igre, le da smo tokrat uporabilo različico 4.0. Poleg tega smo uporabili še knjižnici XEN, ki je grafični API za XNA, in JigLibX za fiziko. Za lažjo distribucijo verzij med člani, smo uporabili Google-ov svn Google Code, kjer so se beležile tudi vse spremembe med verzijami.

Razdelitev dela

Toše Dimkov:
- izdelava menijev

Črtomir Drofenik:
- sistem bojevanja
- animacije modelov
- različna 2D grafika
- inventar
- nasprotniki
- loading screen
- zvoki

Žiga Špindler:
- izrisovanje labirinta
- nalaganje modelov
- preklapljanje med različnimi stanji igre
- premikanje Tezeja
- zajemanje screenshotov
- zaznavanje trkov

Potek dela

Toše Dimkov

Izdelava menijev

Črtomir Drofenik

Sistem bojevanja

Sistem bojevanja smo tokrat naredili potezni. Torej Tezej in njegov nasprotnik sta vsak na svoji strani in lahko vsakič, ko preteče določen čas, izvede eno akcijo. Te akcije so napad(napade nasprotnika), obramba(cel čas se brani, kar pomeni, da mu nasprotnik v tem času naredi manj škode), magija(izvede poseben magičen napad, ki naredi več škode) in inventar(tukaj lahko izbere pobrane elemente, ki mu dajo določene prednosti).

Boj se konča, ko Tezeju ali nasprotniku zdravje pade na 0. Če zmaga Tezej, se ta vrne v labirint in se igra nadaljuje, če pa zmaga nasprotnik, pa se igra konča in se igralec vrne v osnovni meni.

Animacije modelov

Modele Tezeja, Minotavra in orožij nam je posredoval sošolec Blaž Krpan. Ti modeli so vsebovali tudi animacije, ki jih je potrebno samo pravilno klicati v programu. Model Goblina smo pa dobili na spletu. Tako je bilo potrebno samo preveriti, kdaj so katere animacije potrebne. Recimo, ko se premika Tezej, da se sproži animacija za hojo itd. Prav tako je bilo potrebno narediti, da se Modeli obrnjeni v pravo smer, torej, če hodi Tezej v levo, je tako tudi obrnjen.

Različna 2D grafika

V igro je bilo potrebno vključiti različne slike za prikaz stanja Tezeja, inventarja, itd. Prav tako je bilo potrebno tudi narediti ozadje menija in sam izgled galerije.

Inventar

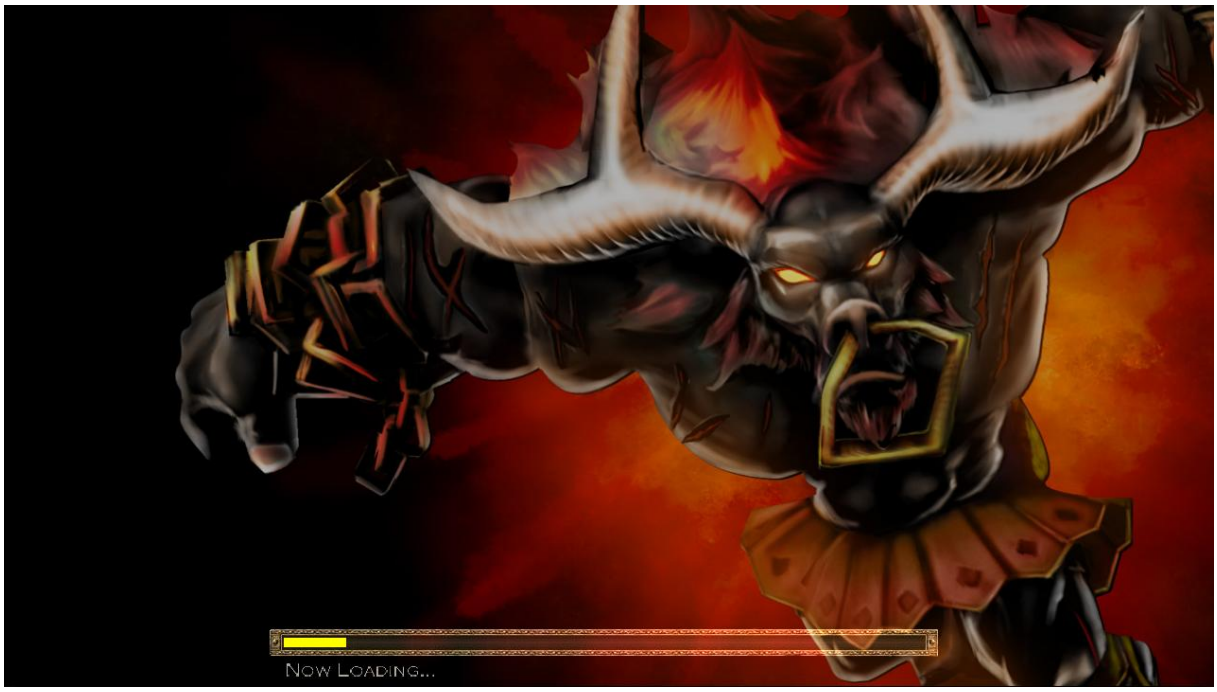
Za inventar je bil narejen razred inventory, v katerem so shranjeni podatki, kaj ima Tezej pri sebi. Prav tako so bile dodane funkcije za izris samega inventarja, kot tudi funkcije za dodajanje in odstranjevanje predmetov.

Nasprotniki

Seveda bi bila igra brez nasprotnikov dolgočasna, zato se je naredil razred enemy, ki je vseboval vse ključne podatke za nasprotnika kot so položaj, moč, zdravje in druge. Nasprotniki se, ob ustvarjanju nove igre, naključno postavijo po labirintu, kar prepreči, da bi bila igra predvidljiva.

Loading screen

Po vključitvi večine funkcij, smo ugotovili, da se igra kar nekaj časa nalaga, zato je bilo potrebno narediti prikaz nalaganje igre, da uporabnik ne bi mislil, da je igra zablokirala.



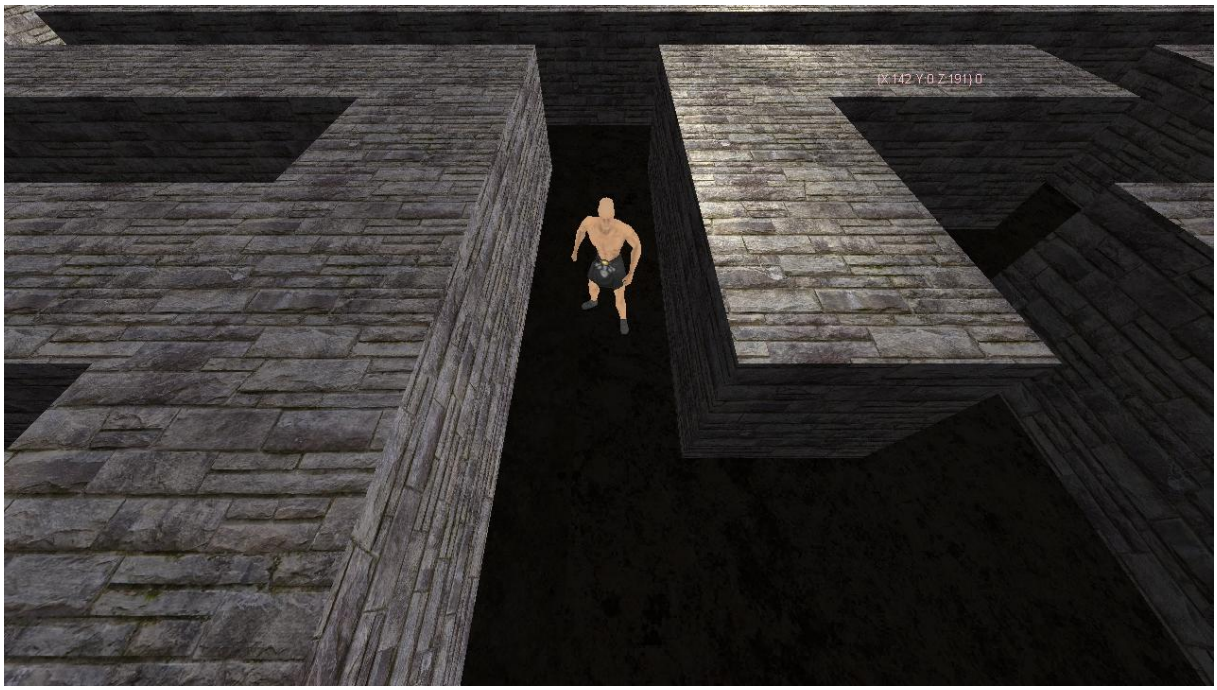
Zvoki

Za boljše vzdušje je bila dodana glasba v ozadju, ki se v spopadu razživi.

Žiga Špindler

Izrisovanje labirinta

Labirint se prebere iz datoteke, kjer je podana velikost in oblika labirinta. Za tla se je uporabila preprosta ploskev s teksturo, za zidove pa kocke, ki imajo prav tako teksture.



Nalaganje modelov

Modele, ki smo jih dobili, je bilo seveda potrebno naložiti v samo igro. To je z XEN API-jem prav enostavno, saj se le pokliče Load metoda za nalaganje datotek iz Content-a.

Preklapljanje med različnimi stanji igre

Za preklapljanje med stanji, kot so meni, igra in bojevanje, se je naredil razred GameStateManager, ki obravnava razrede tipa GameState. GameStateManager poskrbi, da se na zahtevo zamenja sanje in se prejšnje uniči, da se sprostijo strojni viri. GameStateManager je zelo olajšal samo programiranje, saj je sedaj imelo vsako stanje svoje metode za risanje, posodabljanje, nalaganje, itd.

Premikanje Tezeja

Premikanje Tezeja smo naredili s tipkami w,a,s in d. Dokler je tipka pritisnjena, se Tezej premika v želeno smer razen, če je na poti kakšna ovira. Če Tezej naleti na sovražnika, se igra avtomatsko prestavi v bojevanje.

Prav tako je narejeno, da kamera vedno sledi Tezeju, da uporabnik ne rabi iskati Tezeja.

Zajemanje screenshotov

Celotna grafika se sproti riše na teksturo in se šele potem prenese na ekran. To je zelo poenostavilo zajemanje screenshotov, saj sedaj samo pretvorimo tisto teksturo v enega izmed standardnih formatov za slike. Mi smo se odločili za format png.

Zaznavanje trkov

Prvoten načrt je bil, da bi vključili vse prednosti knjižnice JigLibX, a smo kasneje ugotovili, da je bo enostavneje, če uporabimo le »collision detection«, saj je računanje celotne fizike porabilo preveč strojnih virov v primerjavi s tem, kar je doprineslo k igralnosti.

Tako sedaj le gledamo, če se Tezej kje dotakne kakšnega izmed zidov, in v primeru, da se, ga prestavimo na položaj pred trkom.