UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Robert Žagar

Uporaba ogrodja Microsoft XNA z vidika neodvisnega razvijalca iger

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Peter Peer

Asistent: as. Bojan Klemenc

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

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: 00525 / 2013
Datum: 14.9.2013

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

Kandidat: **ROBERT ŽAGAR**

Naslov: **UPORABA OGRODJA MICROSOFT XNA Z VIDIKA NEODVISNEGA RAZVIJALCA IGER**
USE OF MICROSOFT XNA FRAMEWORK FROM INDEPENDENT GAME DEVELOPER PERSPECTIVE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Kadar se lotevamo razvoja igre kot neodvisni razvijalec, poskušamo uporabiti ogrodja ali pogone, ki nam čim bolj olajšajo izdelavo in so brezplačna. Eno takšnih ogrodij je Microsoft XNA. Predstavite ogrodje Microsoft XNA: kako je sestavljeno, kakšna je arhitektura in kako si z njim pomagamo pri izdelavi igre. Preglejte in na kratko opišite tudi možne alternative. Naredite demonstracijsko igro s pomočjo Microsoft XNA in predstavite, kako ste pri tem uporabili ogrodje. Opišite prednosti in omejitve uporabe ogrodja XNA.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Robert Žagar,

z vpisno številko 63040352,

sem avtor diplomskega dela z naslovom:

Uporaba ogrodja Microsoft XNA z vidika neodvisnega razvijalca.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Petra Peera in as. Bojana Klemenca, univ. dipl. inž.,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Petru Peeru in as. Bojanu Klemencu za pomoč pri izdelavi diplomske naloge. Zahvaljujem se tudi svoji družini in vsem drugim, ki so mi kakorkoli pomagali med mojim študijem.

Kazalo

Seznam uporabljenih kratic in simbolov

- 2D (angl. Two-dimensional) – dvorazsežen
- 3D (angl. Three-dimensional) – trirazsežen
- GUI (angl. Graphical User Interface) – grafičen uporabniški vmesnik
- MVP (angl. Microsoft Most Valuable Professional) – naziv, ki ga Microsoft podeli posameznikom, ki so voditelji na področju teoretičnega in praktičnega poznavanja aplikacij Microsoft in svoje znanje delijo z drugimi v obliki člankov oziroma tehnične pomoči
- RTS (angl. Real Time Strategy) – realnočasovna strategija
- FPS (angl. First Person Shooter) – prvoosebna streljačina
- API (angl. Application Programming Interface) – programski vmesnik
- AI (angl. Artificial Intelligence) – umetna inteligenca
- CPU (angl. Central Processing Unit) – centralna procesna enota
- GPU (angl. Graphical Processing Unit) – grafična procesna enota
- OS (angl. Operating System) – operacijski sistem
- XACT (angl. Cross-platform Audio Creation Tool) – večplatformska zvočna knjižnica in pogon, ki jo je izdal Microsoft
- RPG (angl. Role-Playing Game) – igra igranja vlog
- IDE (angl. Integrated Development Environment) – integrirano razvojno okolje

- CLR (angl. Common Language Runtime) – izvajalnik, skupen vsem programskim jezikom
- VM (angl. Virtual Machine) – navidezni stroj
- EULA (angl. End-User License Agreement) – licenčna pogodba s končnim uporabnikom
- HLSL (angl. High Level Shader Language) – visokonivojski jezik za senčenje

Povzetek

Ko se lotimo razvoja nove računalniške igre, nam zelo pomaga, če lahko začnemo razvoj na že obstoječem ogrodju oziroma pogonu, ki nam prihrani programiranje najbolj osnovnih funkcij, ki jih potrebujemo v vsaki igri. V diplomskem delu si ogledamo podrobnosti o brezplačnem Microsoftovem ogrodju XNA in nekaj uspešnih komercialnih iger, ki so jih izdelali neodvisni razvijalci z uporabo tega ogrodja. Na kratko si pogledamo tudi nekaj drugih brezplačnih ogrodij, pogonov in orodij, ki jih neodvisni razvijalci lahko uporabijo za razvoj računalniških iger. S pomočjo ogrodja XNA tudi realiziramo demonstracijsko igro in v njej demonstriramo funkcije tega ogrodja.

Ključne besede:
ogrodje XNA, Microsoft, igra, programiranje, neodvisni razvijalci

Abstract

When we start to develop a new computer game it helps us a lot if we start the development with an existing framework or game engine, because it spares us the programming of the most basic functions that we need in every game. In this thesis we present details about a free Microsoft XNA Framework and a few successful commercial games that were made by indie developers using this framework. We also list a few other free frameworks, game engines and tools that can be used by indie developers to develop computer games. With the help of XNA Framework we also realize a demonstration game and in it demonstrate functions of this framework.

Key words:
XNA Framework, Microsoft, Game, Programming, Indie developers

Poglavje 1

Uvod

Video igre so prišle zelo daleč od njihovih začetkov, ko so jih razvijali posamezniki ali manjše skupine in niso vsebovale nič oziroma zelo malo multimedijskih vsebin. V današnjem času se v razvoj in promocijo komercialnih iger vlagajo milijonski zneski, te vsebujejo veliko multimedijskih vsebin, kot so glasba, filmi, zvočni učinki itd. Igre izdelujejo velika podjetja, ki zaposljujejo grafične oblikovalce, programerje, tonske mojstre, skladatelje, prevajalce itd. Posamezniki in manjše skupine ne morejo konkurirati tem velikim podjetjem v količini in kakovosti multimedijskih vsebin, zato se velikokrat zanašajo na inovativnost in možnost modifikacije iger s strani igralcev. Tem razvijalcem pravimo samostojni razvijalci. V zadnjem času opažamo precejšen porast števila samostojnih iger zaradi novih načinov razpečevanja preko spleta (na primer Valve-ov Steam) in novih brezplačnih oziroma cenejših razvojnih orodij, ogrodij ter igralnih pogonov.

Microsoft XNA je eno teh razvojnih ogrodij, s katerim lahko popolnoma brezplačno izdelamo komercialne video igre za Microsoft Windows, Xbox 360 in Windows Phone 7. Cilj tega ogrodja je razvijalca osvoboditi pisanja ponavljajoče se kode in spraviti različne vidike razvijanja računalniških iger v en sam sistem [1]. Izvorno kodo lahko z manjšimi spremembami oziroma celo brez njih prevedemo za različne podprte platforme, saj se koda piše na visokem nivoju. Izdelane igre in aplikacije za Xbox in Windows Phone se lahko preprosto prodajo preko Microsoftovega digitalnega spletnega portala App Hub, kjer lahko najdemo tudi velik izbor izobraževalnih vsebin za delo z Microsoft XNA.

Glavna tema diplomske naloge je predstavitev Microsoft XNA in pregled prednosti ter slabosti uporabe tega razvijalskega ogrodja. Te teme so zajete v poglavjih od 2 do 4.

V petem poglavju je na kratko opisanih nekaj bolj znanih komercialnih iger, ko so bile izdelane z Microsoft XNA.

V šestem poglavju lahko vidimo, kako nam ogrodje XNA pomaga pri izdelavi računalniške igre na konkretnih primerih iz demonstracijske igre.

V sedmem poglavju so opisane demonstracijska igra, njene komponente in nekatere zanimivosti.

Sedmo poglavje vsebuje zaključek.

Poglavje 2

Predstavitev Microsoft XNA

2.1 Kratica XNA

Kratica XNA izvira iz razvojnega imena projekta - Xbox New Arhitecture. Ogrodje ni bilo izdano pod tem imenom, ker je bil med razvojem ogrodja izdan novejši Xbox 360. Prva verzija pa je imela tudi podporo za Microsoft Windows. Kratica XNA naj bi pomenila ,,XNA is Not an Acronym“, kar pa v prevodu pomeni ,,XNA ni kratica“ [2].

2.2 Opis

Microsoft XNA je zbirka orodij za razvoj računalniških iger za Microsoft Windows, Xbox 360 in Windows Phone 7. Glavna cilja pri zasnovi XNA sta bila zagotoviti prenosljivost izvorne kode med platformo Windows in platformo Xbox ter poenostavitev razvoja iger [3, 4, 5].

Običajno razvijalec potrebuje veliko kode, časa, poskusov in napak, preden lahko začne delati na sami vsebini igre - z XNA lahko začnemo delati na vsebini igre takoj. Obširna dokumentacija, primeri, tečaji in začetniški paketi omogočajo uporabnikom enostaven in hiter razvoj 2D- in 3D-iger.

Razvoj komercialnih iger za Windows je brezplačen, vendar se igre ne smejo povezovati na Xbox Live oziroma Games For Windows Live brez dogovora med podjetjem Microsoft in založnikom oziroma razvijalcem (več o tem, kaj ponujajo storitve Xbox Live in Games For Windows Live, si lahko ogledamo v podpoglavju 2.4.2). Lahko pa razvijalec implementira večigralsko funkcionalnost preko drugih knjižnic oziroma lastne izvedbe. Če želi razvijalec prodajati svoj izdelek za Xbox 360 ali Windows Phone 7 preko Mi-

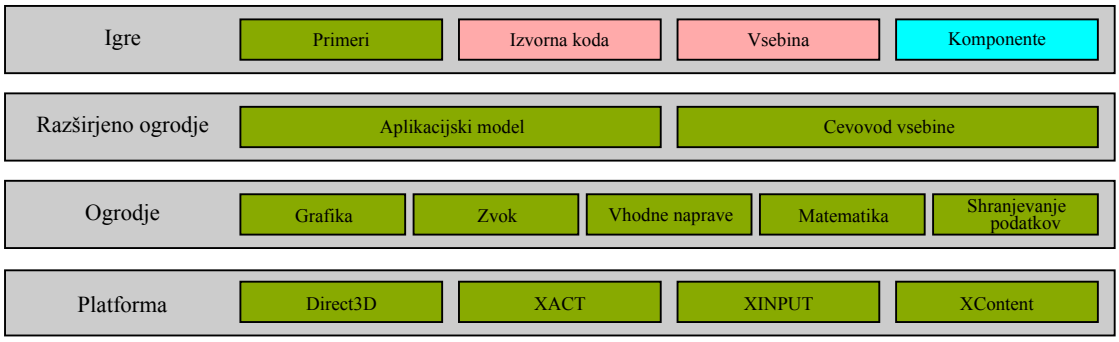
crosoftovega portala App Hub oziroma Windows Phone Marketplace, mora plačati 99 dolarjev letne naročnine in je upravičen do 70 % prihodkov, ki jih Microsoft zasluži z igro oziroma aplikacijo [1].

Zbirko sestavljajo ogrodje XNA (angl. XNA Framework), XNA Build in XNA Game Studio. Podrobnosti posameznih komponent si bomo ogledali v nadaljevanju.

2.2.1 Ogrodje XNA

Ogrodje XNA je zasnovano na nativni implementaciji .NET Compact Framework 2.0 za Xbox 360 in .NET Framework 2.0 za Windows. Vključuje obširno zbirko knjižnic, ki so specifične za razvoj računalniških iger in zagotavljajo maksimalno prenosljivost med podprtimi platformami. Ogrodje teče na modificirani verziji Common Language Runtime, ki je optimizirana za igre in zagotavlja upravljano izvajalno okolje. Izvajalno okolje XNA je na voljo za Windows XP, Windows Vista, Windows 7, Windows Phone and Xbox 360. XNA igre so napisane za izvajalno okolje, kar zagotavlja, da jih lahko uporabljamo na vseh platformah, ki podpirajo ogrodje XNA, z minimalnimi modifikacijami oziroma brez njih [1].

Igre so lahko napisane v poljubnem programskem jeziku .NET, toda samo C# v XNA Game Studio Ekspress in vseh verzijah Visual Studio 2008 in 2010 je uradno podprt. Leta 2011 je bila dodana še podpora za Visual Basic .NET [1].



Slika 2.1: Arhitektura ogrodja XNA [6] - V rdečem odtenku so obarvani razdelki, ki jih izdela uporabnik ogrodja XNA. V modrih odtenkih so razdelki, ki so jih izdelali drugi uporabniki in jih lahko uporabimo v svojih projektih. V zelenih odtenkih pa so razdelki, ki so vključeni z ogrodjem XNA.

Ogrodje XNA enkapsulira nizkonivojske tehnične podrobnosti pri razvi-

janju iger in tako poskrbi za razlike med različnimi podprtimi platformami ter omogoči razvijalcem, da se osredotočijo bolj na samo vsebino igre in igralnost. Arhitekturo ogrodja XNA lahko vidimo na sliki 2.1. Posamezne elemente arhitekture si bomo ogledali v nadaljevanju in tudi v drugih poglavjih. Framework podpira razvoj 2D- in 3D-iger, uporabo igralnih pripomočkov Xbox 360 in ponuja še razna druga orodja, kot na primer XACT - cross platform Audio Creation Tool [1].

Microsoft XNA nam ponuja [5]:

- Aplikacijski model - namen katerega je abstrahirati platformo, na kateri teče igra in tako zagotoviti, da se lahko osredotočimo na pisanje igre. Ni nam treba skrbeti za kreacijo okna, upravljanje s sporočili oken in tem kakšno grafično enoto ima računalnik. Tukaj je implementiran tudi model komponent, ki nam omogoča hitro implementacijo komponent, napisanih s strani drugih razvijalcev v naših igrah.

- Grafika - grafični API ogrodja XNA je baziran na Direct3D API, vendar je njegova uporaba veliko enostavnejšsa in bolj razumljiva. Večina 3D-grafike je zasnovana na principu učinkov, senčenja in programljivega cevovoda.

- Zvok - zvočni API ogrodja XNA je baziran na XACT, katerega ideja je enostavno delo z zvokom v igrah. Naložiš zvočno datoteko in nastaviš glasnost, ponavljanje itd. Za vse stvari v ozadju (inicializacija bufferjev, branje itd.) pa poskrbi API.

- Vhodne naprave (miška, tipkovnica itd.) - tudi tukaj je vse narejeno tako, da je čimbolj enostavno za uporabo. Ni potrebne nobene inicializacije, iskanja vhodnih naprav ali nastavljanja posebnih nastavitev. Samo uporabimo metodo GetState() na želeni napravi in dobimo njeno trenutno stanje (kateri gumbi so pritisnjeni na tipkovnici, trenutna pozicija miške itd.).

- Shranjevanje podatkov - na platformi Microsoft Windows s tem ni posebnih težav, ker pa mora XNA biti kompatibilna z več platformami naenkrat, vsebuje metode, s katerimi lahko shranjujemo podatke neodvisno od platforme, na kateri teče igra.

- Matematika - XNA vsebuje veliko tipov, ki so zelo priročni pri programiranju iger (Vector2, Vector3, Vector4, Matrix, Plane, Ray, BoundingBox, BoundingSphere, itd.), in tudi metode, ki nam prihranijo veliko

matematike. Kot primer lahko vzamemo metodo CreateLookAt(), s katero je delo s 3D-kamero mnogo lažje.

2.2.2 XNA Build

XNA Build je komplet orodij za upravljanje s cevovodom vsebine igre, ki pomaga z definiranjem, vzdrževanjem, debugiranjem in optimizacijo cevovoda pri posameznem projektu. Cevovod vsebine igre definira proces, ki modificira vsebino (3D-modele, teksture itd.) v obliko, ki jo razume igralni pogon. XNA Build pomaga identificirati odvisnosti cevovoda in omogoča API dostop do podatkov o odvisnosti med vsebino in kodo. Te odvisnosti lahko analiziramo in tako poskušamo najti vsebino, ki ni nikjer v kodi v uporabi [1].

Analiza igre MechCommander 2 s tem orodjem je na primer odkrila, da kar 40 % tekstur, ki so bile vključene v igri, niso bile v uporabi in bi jih lahko odstranili [1].

2.2.3 XNA Game Studio

XNA Game Studio je integrirano razvijalsko orodje, ki nam omogoča lažji razvoj računalniških iger za podprte platforme. Game Studio je pravzaprav nadgradnja Microsoft Visual Studia s podporo za ogrodje XNA, z novimi predlogami in orodij za uvoz zvočne ter grafične vsebine igre. Tako lahko uporabljamo funkcionalnost ogrodja XNA za pisanje kode, specifične razvoju iger (grafični prikaz, uporabnikovi vhodni podatki itd.) in .NET Framework za bolj splošne programerske naloge [1].

2.3 Zakaj Microsoft XNA

Temo te diplomske naloge sem izbral zaradi svojih predhodnih izkušenj s tem ogrodjem. Pred leti sem bil del skupine, ki je razvijala neprofitno masivno večigralsko igro in je za razvoj te uporabljala ogrodje Microsoft XNA ter dodatne brezplačne knjižnice.

V nadaljevanju si oglejmo še nekaj bolj splošnih razlogov:

- Vse, kar potrebujemo, je brezplačno - Microsoft Visual Studio Ekspress in XNA Game Studio sta popolnoma brezplačna za uporabo.
- Enostavnost - takoj se lahko posvetimo razvoju igre, namesto da zapravimo veliko časa pri pisanju podporne kode.

- Obširna dokumentacija in primeri - na medmrežju so nam na voljo obširna dokumentacija, video tečaji, baza primerov in celo tako imenovani začetniški paketi, ki so praktično celotne igre ter jih lahko modificiramo in uporabljamo po želji tudi v lastnih projektih.

- Podprtih več popularnih igralnih platform - danes je večina največjih iger izdana na platformah Windows, Xbox 360 in Playstation 3. Z Microsoft XNA imamo torej podporo za dve od treh največjih igralnih platform in tudi podporo za razvoj iger za mobilne telefone z operacijskim sistemom Windows Phone 7.

Obstajajo številne alternative, ki jih lahko uporabimo namesto Microsoft XNA. Vsaka od njih ima svoje prednosti in slabosti. Nekaj alternativnih možnosti si bomo pogledali v kasnejših poglavjih.

2.4 Zgodovina

Microsoft XNA je bil prvič omenjen leta 2004 na GDC (Game Developers Conference). Bilo je veliko ugibanj glede tega, kaj točno je XNA in ali bo popolnoma zamenjal knjižnico DirectX, toda ko je bil leta 2005 izdan Direct3D 10 za Windows Vista, je bilo videti, da bo DirectX glavna grafična knjižnica tudi vnaprej. Potem vse do leta 2006 ni bilo o XNA slišati nič. Na začetku leta 2006 je bil na GDC izdan XNA Build. Sledilo je nekaj mesecev tišine in samo Microsoftovi uslužbenci ter MVP so vedeli o prihajajočih izdajah ogrodja XNA in XNA Game Studio. Preostali svet je za to izvedel na Gamefest conference, kjer je Microsoft oznanil, da bo XNA Game Studio Ekspress beta 1 izdan 30. avgusta leta 2006. Nekaj mesecev pozneje je bila izdana še ena beta različica, decembra 2006 pa končna različica - Game Studio Ekspress 1.0 [1, 2].

2.4.1 Game Studio Ekspress 1.0

Prva izdaja, namenjena predvsem tistim, ki se z razvojem iger ukvarjajo za hobi - študenti, neodvisni razvijalci itd. Vsebovala je začetniške pakete za hiter razvoj iger specifičnih žanrov, kot so RTS, FPS in arkadne igre [2].

Na začetku leta 2007 je bila izdana še posodobitev z imenom Game Studio Ekspress 1.0 Refresh, ki je upoštevala veliko želj uporabnikov ter odpravila nekaj hroščev [2].

2.4.2 Game Studio 2.0

Konec leta 2007 je bil izdan Game Studio 2.0, ki je vseboval veliko izboljšav. Najbolj pomembne izboljšave:

- možnost uporabe z vsemi verzijami Visual Studio 2005 - tudi z Ekspress Edition,
- večigralski API, ki uporablja LIVE na platformah Windows in Xbox 360 [7, 1].

Microsoft Xbox Live in Games for Windows Live so spletne storitve, ki ponujajo [8, 9]:

- uporabniški profil z informacijami o vsakem igralcu, katere igre igra, kdo so njegovi prijatelji itd.,
- zaslužek točk z odklepanjem dosežkov v igrah,
- možnost klepetanja s prijatelji,
- igranje večigralskih iger preko interneta,
- izbiro nasprotnikov v večigralskih igrah glede na izkušenost igralca, regijo, nacionalnost itd.,
- možnost igranja večigralskih iger proti nasprotnikom z druge platforme (Xbox 360, Windows), če igra to podpira,
- storitve so tudi moderirane in uporabniki proti katerim je vloženih veliko pritožb (zaradi sovražnega govora, goljufanja itd.), so kaznovani,
- glasovno in video komunikacijo.

2.4.3 Game Studio 3.0

Oktobra 2008 je bila izdana končna različica Game Studio 3.0. Spisek izboljšav razvojnega okolja je dolg, tako da si bomo ogledali samo pomembnejše novosti [10, 1], ki so:

- podpora za Visual Studio 2008,
- podpora za C# 3.0,

- možnost razvoja iger za Microsoft Zune,
- izboljšana podpora za razvoj iger, ki ciljajo na več od podprtih platform,
- ClickOnce Deployment za Windows, podrobnosti te opcije si bomo ogledali kasneje,
- transparentna kompresija in dekompresija vsebine igre na platformah Windows in Xbox 360.

2.4.4 Game Studio 3.1

Poleg številnih manjših popravkov in nove funkcionalnosti je Game Studio 3.1 vseboval tudi naslednje pomembnejše novosti [11, 1]:

- podporo za Xbox 360 Avatars
- podporo za Xbox Live Party
- možnost predvajanja video vsebin

2.4.5 Game Studio 4.0 in Game Studio 4.0 Refresh

Game Studio 4.0 Refresh je trenutno aktualna verzija. Vsebuje veliko izboljšav in popravkov, toda glavna novost je podpora za platformo Microsoft Windows Phone 7, prav tako pa je nova večina metod, namenjenih za delo s to platformo [12]. Ena pomembnejša novost, ki ni povezana z Windows Phone 7 in je vredna omembe, je podpora za programski jezik Visual Basic [13].

Poglavje 3

Prednosti in slabosti uporabe Microsoft XNA

3.1 Prednosti

3.1.1 Splošno

O večini prednosti uporabe Microsoft XNA smo govorili že v prejšnjih poglavjih, tako da jih bomo samo povzeli:

- Uporaba XNA je brezplačna - v naslednjih poglavjih si bomo ogledali tudi nekaj alternativ, ki so brezplačna za uporabo. Videli pa bomo tudi, da je, medtem ko je razvoj iger brezplačen, treba za nekatere platforme plačati naročnino, če želimo igro prodajati.

- Podprtih je več popularnih igralnih platform - v 5. poglavju si bomo pogledali nekaj alternativnih oziroma konkurenčnih igralnih pogonov in ogrodji, ki podpirajo še veliko več platform (toda večina teh je namenjena bolj profesionalni uporabi in tako niso zelo zanimive kot ciljne platforme za komercialne igre).

- Obširna dokumentacija, primeri, vodiči so na voljo tako na uradni spletni strani kot tudi na mnogih drugih spletnih straneh - mnoge alternative imajo zelo aktivne skupnosti, ki pomagajo uporabnikom in ponavadi so na voljo poleg API-dokumentacije tudi vodiči o osnovni uporabi pogona oziroma ogrodja. Toda Microsoft XNA ima na uradni strani:

- na stotine kratkih vzorcev kode, ki demonstrirajo, kako se implementira želena funkcionalnost v igro,

- običajne in video vodiče, ki demonstrirajo, kako izdelati 2D- ali 3D-igro v celoti na enostavnih primerih,

- aktivno skupnost,

- izvorno kodo za nekaj iger, izdelanih v XNA (lahko uporabimo za osnovo za izdelavo lastnih iger oziroma preučimo izvorno kodo, da vidimo, kako deluje).

V nadaljevanju so opisane še nekatere druge prednosti in tudi že omenjene prednosti z dodatnimi informacijami.

3.1.2 Cevovod vsebine (angl. Content Pipeline)

Vsebina je pomemben del današnjih iger in uporaba več različnih formatov vsebine v igri je lahko zelo problematična. Na srečo nam tukaj pomaga ogrodje XNA s cevovodom vsebine, ki stvari zelo poenostavi. Več podrobnosti v nadaljevanju.

3.1.2.1 Vsebina (angl. Content)

Z besedo vsebina bomo v nadaljevanju označevali teksture, slike, grafične učinke, pisave, zvočne učinke, glasbo, 3D-modele in tudi podatkovno vsebino, kot so razne tabele stopenj, atributov in podobno. Vse te stvari so lahko sestavni del vsebine igre.

Različnih orodij za izdelavo vsebine je veliko in vsako orodje ponavadi shranjuje vsebino v svojem posebnem formatu. Zato imamo lahko velike probleme z iskanjem raznih pretvornikov med različnimi formati, izvoznih oziroma uvoznih skript in orodij, ki podpirajo format, ki ga potrebujemo. Poleg tega moramo vsebino še obdelati in opraviti vse potrebno za uporabo te v igri. Veliki razvojni studiji imajo včasih cele ekipe ljudi, ki se ukvarjajo samo z izgradnjo te infastrukture [14].

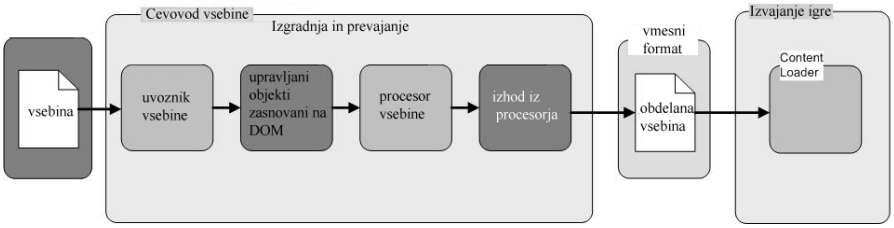
3.1.2.2 Cevovod vsebine

Izdelava infastrukture za upravljanje in delo z vsebino je še en del podporne kode in cevovod vsebine nam ta del poenostavi ter omogoči večjo svobodo pri izbiri orodij za izdelavo vsebine in delo z njo. To nam omogoči s podporo za veliko različnih formatov, ki so bili izbrani, ker zna z njimi delati

večina pomembnejših orodij za delo z vsebino. Lahko tudi prilagodimo način uvoza in obdelave posameznih formatov ter tudi dodamo podporo za še več formatov [14].

Dodatna prednost uporabe cevovoda vsebine je hitrejše nalaganje vsebine med izvajanjem igre. Brez cevovoda vsebine mora igra ugotoviti, v kakšnem formatu je vsak del vsebine, in ga pretvoriti v obliko, ki jo lahko uporabi. Kot vidimo na sliki 3.1, cevovod vsebine pretvori vsak del vsebine v ustrezen format med povezovanjem, izgradnjo in prevanjanjem izvorne kode, tako igri tega ni treba narediti med izvajanjem [3, 15].

Novo vsebino dodajamo na enak način, kot če bi projektu dodali novo datoteko z izvorno kodo - to je z uporabo razvojnega orodja Game Studio. Tako imamo celoten razvoj igre na enem samem mestu, kjer ga lahko organiziramo po svojih željah. Ko dodamo novo vsebino, nastavimo, kako naj jo cevovod vsebine uvozi, obdela in upravlja. Organizacijo, uporabo in nastavitve uvoznika ter procesorja lahko vidimo na sliki 3.2.

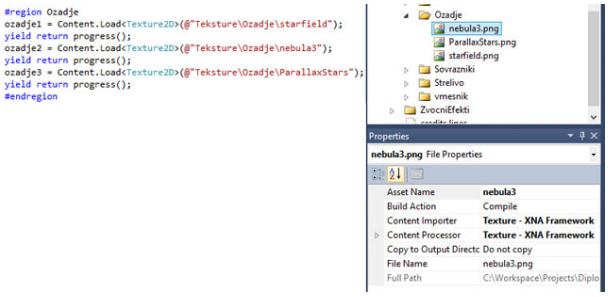


Slika 3.1: Diagram prikazuje kako deluje cevovod vsebine [16]

Cevovod vsebine torej poskrbi za uvoz in pretvorbo vsebine v ustrezno obliko oziroma jo prekopira v mapo, kjer jo potrebujemo. Tukaj pa ni mišljena samo vsebina, kot so teksture, 3D-modeli in ostale podobne oblike vsebine, ampak tudi poljubna vsebina, kot so na primer tekmovalne proge v igri, kjer tekmujemo z avtomobili ali datoteke xml z vsebino razvojnega drevesa v strateški igri, itd. [14].

3.1.3 Ciljne platforme

Microsoft XNA ne samo, da podpira več ciljnih platform, ampak omogoča sočasen razvoj iger za vse platforme, ki jih podpira.



Slika 3.2: Primer organizirane vsebine igre v Game Studio in njene uporabe

3.1.3.1 Podpora za več popularnih igralnih platform

Dandanes je večina komercialnih iger izdanih na treh platformah, to so Microsoft Windows, Xbox 360 and Playstation 3. Kot vidimo, z razvojem iger z uporabo Microsoft XNA dosežemo dve od teh treh platform brez dodatnega dela s prenosom igre na drugo platformo.

Omeniti pa treba tudi platformo Windows Phone, saj se trg za mobilne igre in aplikacije za to platformo povečuje [17]. Spletna trgovina je dostopna že kar v 54 državah, med katerimi je od leta 2012 tudi Slovenija [18].

3.1.3.2 Sočasen razvoj igre za več ciljnih platform

```
#if WINDOWS
// Izvorna koda za platformo Microsoft Windows
#elif XBOX
// Izvorna koda za platformo Xbox
#elif WINDOWS_PHONE
// Izvorna koda za platformo Microsoft Windows Phone
#else
// Vrne napako med prevajanjem: Ta platforma ni podprta.

#endif
```

Razvoj iger v XNA Game Studio je na visokem nivoju in, kolikor je to le mogoče, neodvisno od strojne opreme, na kateri se bo igra izvajala, zato lahko istočasno programiramo igro za vse podprte platforme. Nekatere razlike med različnimi podprtimi platformami pa je treba še vedno upoštevati tudi v izvorni kodi. Kako to dosežemo, prikazuje zgornji primer. Te razlike so

ponavadi v igralnih pripomočkih (tipkovnica in miška na Windows, kontroler „Gamepad“ na Xbox 360) ali grafičnih zmogljivostih (Windows Phone v primerjavi z Xbox 360 ali PC) [19, 20].

3.1.4 Enostavna in razumljiva osnovna struktura izvorne kode

Ko kreiramo nov projekt, nam XNA odpre enostavno osnovno predlogo (game.cs) z metodami, ki jih bomo potrebovali. Te metode si bomo ogledali v nadaljevanju [3].

- Initialize - metoda se izvrši, preden se karkoli izriše na zaslonu. V njej preverimo prisotnost potrebnih komponent in naložimo vsebino, ki ni grafične narave. Tukaj tudi nastavimo parametre za grafični prikaz (resolucija, glajenje robov, itd.).

- Loadcontent - tukaj naložimo vso grafično vsebino igre, kot so 3D-modeli, teksture, pisave itd.

- Unloadcontent - ta metoda se izvrši pred izhodom iz igre in je namenjena čiščenju vsebine ter komponent, ki smo jih naložili brez uporabe Content Manager-ja.

- Update - ta metoda je poleg Draw del igrine zanke in se izvršuje 60-krat na sekundo (v kolikor ni preobremenjena in uporabimo privzete nastavitve). V njej se izvaja vsa logika. Na primer: branje vhodnih podatkov (tipkovnica, miška itd.), detekcija trkov med 3D-modeli, premikanje 3D-modelov, AI itd.

- Draw - metoda Draw poskrbi za prikaz igre na zaslonu.

3.1.5 Namestitev z enim klikom

Glavna naloga tehnologije namestitev z enim klikom (angl. ClickOnce Deployment) je hitra in enostavna namestitev aplikacije na platformi Microsoft Windows. Med namestitvijo aplikacija avtomatsko poišče, prenese s spleta in namesti vse dodatne komponente, ki so potrebne za delovanje aplikacije. Pri namestitvi je potrebna minimalna uporabnikova interakcija.

Poleg vsega tega nam ta tehnologija ponudi rešitve za tri probleme pri razpečavi aplikacij [21, 22]:

- Posodabljanje aplikacije - z uporabo standardne namestitve, ki je na voljo v Visual Studio (angl. Windows Installer Deployment) moramo aplikacijo ob posodobitvi izbrisati in ponovno namestiti; Pri namestitvi z enim klikom je posodabljanje avtomatizirano in s spleta se prenesejo le tisti deli aplikacije, ki so bili posodobljeni.

- Učinek na uporabnikov računalnik - aplikacije velikokrat potrebujejo dodatne komponente specifične verzije in jih zato inštalirajo, kar lahko privede do težav z drugimi aplikacijami, ki so potrebovale drugo verzijo iste komponente; S to tehnologijo aplikacije ne morejo vplivati na druge aplikacije na uporabnikovem računalniku.

- Varnostna dovoljenja - aplikacije mnogokrat potrebujejo administratorske pravice za namestitev; S to tehnologijo jih ne potrebujejo in se namestijo le za uporabo trenutnega uporabnika.

3.2 Slabosti

3.2.1 Splošno

V tem poglavju si bomo pogledali še slabosti uporabe Microsoft XNA za razvoj iger.

Najprej na hitro naštejmo slabosti uporabe XNA:

- naročnina za Xbox 360 in določena maksimalna dovoljena velikost igre [1, 23, 24],

- igra, napisana v XNA, je malo počasnejša v primerjavi s tistimi, napisanimi v C++ in DirectX oziroma OpenGL [25],

- ne moremo uporabiti Windows Live oziroma Xbox live [1],

- prenosljivost na nepodprte platforme je kompliciranа,

- predvideva uporabo drugih Microsoft-ovih aplikacij, ki niso brezplačne.

V nadaljevanju bomo videli dodatne informacije o teh in drugih slabostih uporabe ogrodja XNA.

3.2.2 Naročnina za Xbox 360

Igre za Xbox 360 lahko vnesemo v bazo iger na Microsoftovi spletni strani App Hub, toda plačati moramo letno naročnino. Vse igre, dodane na App Hub, so ocenjene s strani drugih članov te spletne strani. Če igra dobi dobre ocene, jo Microsoft prodaja preko Xbox Live Marketplace. Avtor igre določi njeno ceno in dobi 70 % od celotnega prihodka prodaje igre. Določena je tudi maksimalna velikost igre, ki jo lahko avtor doda na stran. Od januarja 2012 je maksimalna velikost 500 MB, pred tem pa je bila 150 MB [1, 23, 24].

3.2.3 Počasnejše izvajanje v primerjavi z nekaterimi alternativami

Pravilo je, da bolj ko programiramo na visokem nivoju, bolj požrešna je naša aplikacija oziroma igra. XNA uporablja upravljano kodo (angl. Managed Code), ki jo mora VM interpretirati. Microsoft s tem izrazom označuje izvorno kodo, ki za izvršitev potrebuje CLR VM. To povzroči počasnejše izvajanje programa v primerjavi z nativno kodo (na primer C++ z DirectX). Poleg počasnejšega izvajanja ponavadi lahko opazimo tudi nekoliko večjo uporabo sistemskih resursov [25].

3.2.4 Ne moremo uporabiti Windows Live oziroma Xbox Live

Microsoft XNA EULA prepoveduje distribucijo komercialnih iger, ki se povežejo na Xbox Live oziroma Games for Windows Live brez dogovora med razvijalcem in Microsoftom. Še vedno lahko z XNA Game Studio razvijemo in izdamo komercialne igre za Microsoft Windows, vendar moramo omrežno kodo napisati sami oziroma moramo uporabiti zunanje knjižnice. Igre za Xbox 360 in Windows Phone 7 moramo izdati preko Microsoftove spletne strani - torej moramo plačati naročnino [1].

3.2.5 Prenosljivost

Medtem ko podpira več platform praktično brez sprememb izvorne kode, je prenos na ostale nepodprte platforme zelo težaven. V zadnjem času se je pojavilo veliko alternativnih implementacij XNA frameworka, ki nam to nalogo

olajšajo (MonoXNA, MonoGame, ANX). Več informacij o teh alternativah si bomo pogledali v naslednjem poglavju [1].

3.2.6 Povezanost z drugimi Microsoftovimi aplikacijami

Najprej seveda potrebujemo Microsoft Windows OS, ki ni brezplačen.

Poleg tega Visual Studio tudi predvideva uporabo podatkovnih baz Microsoft SQL. Microsoft SQL Server Ekspress je sicer brezplačen, vendar sta njegova funkcionalnost in zmogljivost precej okrnjeni. Z uporabo zunanjih knjižnic je možno uporabiti tudi druge brezplačne podatkovne baze (na primer MySql).

Tudi sam Visual Studio Ekspress nima vse funkcionalnosti, ki jih imajo druge izdaje. Pomembnejše funkcije, ki jih v primerjavi z drugimi različicami (Professional, Ultimate itd.) Visual Studio Ekspress nima [26]:

- podpore za dodatke (razne skripte, napisane s strani uporabnikov, ki dodajo dodatno funkcionalnost programu oziroma izboljšajo trenutno funcionalnost),
- profiliranje (z njim vidimo, koliko spomina uporablja naša aplikacija, najdemo ozka grla in optimiziramo aplikacijo),
- 64-bitni prevajalnik,
- vgrajen urejevalnik vsebine (na primer za ikone).

3.2.7 Prihodnost Microsoft XNA in Microsoft Windows 8 Metro

Microsoft je uradno napovedal, da bo ogrodje XNA podpiral le še do aprila 2014. Sporočil je tudi, da nadaljnih posodobitev ne bo in da je verzija 4.0 končna verzija [27].

Trenutna verzija ogrodja XNA ne podpira razvoja iger za Windows 8 Metro. Če želimo izdelati igro za to platformo, moramo uporabiti MonoGame (več o tem v naslednjem poglavju).

Poglavje 4

Kratek pregled alternativnih možnosti

4.1 Igralni pogon, ogrodje, knjižnica

Igralni pogon je sistem, izdelan za razvoj video iger. Glavna komponenta vsakega pogona je grafični sistem. Velikokrat vsebuje še sistem za simuliranje fizičnih učinkov, zvočni sistem, sistem za umetno inteligenco, omrežno podporo itd. Pogon je lahko specializiran za razvoj specifičnega žanra iger. Delo s pogoni ponavadi poteka na zelo visokem nivoju z uporabo vizualnih razvojnih orodij in skriptnih jezikov. Igralni pogon je lahko zgrajen na osnovi ogrodja [28].

Ogrodje nam ponuja bolj splošno funkcionalnost, ki jo selektivno z lastno kodo priredimo potrebam svoje aplikacije. Delo z ogrodji ponavadi poteka na nižjem nivoju kot delo s pogoni. To nam omogoči večjo kontrolo nad funkcionalnostjo aplikacije, vendar porabimo več časa za njen razvoj. Na primer: pogon že vsebuje sistem za delo z animacijami, medtem ko ogrodje vsebuje vse sestavne dele in jih moramo sami povezati v sistem. V nasprotju s knjižnicami ogrodje izvrši našo izvorno kodo in odloča, v kakšnem vrstnem redu se izvajajo metode. Ogrodje lahko vsebuje več knjižnic [29].

Knjižnica je skupek funkcij oziroma metod, ki jih lahko uporabiš kjerkoli v svoji izvorni kodi. Po izvršitvi klicane funkcije se izvajanje aplikacije nadaljuje od tam, kjer je bila klicana. Knjižnice so ponavadi zelo ozko specializirane.

4.2 Brezplačni igralni pogoni

Igralni pogoni so v veliko primerih specializirani za eno samo vrsto iger (na primer 3D-streljačine).

Nekaj pogonov, ki so specializirani:

- Thousand Parsec (osnova za gradnjo poteznih iger v katerih gradiš svoj imperij),
- Cube Engine (3D streljačine),
- itd.

Torej, če izberemo primeren pogon za vrsto igre, ki jo želimo narediti, lahko prihranimo veliko časa. Dobro si je tudi ogledati vse, kar posamezni pogon ponuja, in v katerem programskem jeziku se izvaja končni produkt ter skriptni jezik, ki ga je treba znati za pisanje kode.

Igralni pogoni so velikokrat tudi zaprtokodni in tudi, kadar niso so zelo obsežni in kompleksni ter jih je zato težko modificirati brez veliko priučevanja in analiziranja, kar nam lahko vzame veliko časa.

Formati vhodnih datotek so pri posameznem pogonu ponavadi določeni in potrebujemo računalniške programe za urejanje in izdelavo datotek teh formatov. Sledi, seveda, da moramo tudi znati delati s temi programi.

V nadaljevanju si bomo malo podrobneje ogledali, kaj nam ponuja par bolj znanih igralnih pogonov, ki so bili uporabljeni za izdajo komercialnih iger.

Panda3D

Poleg grafičnega sistema vsebuje še [30]:

- zvočni sistem,
- vmesnik za tipkovnico, miško itd.,
- sistem za detekcijo trkov,
- sistem za simuliranje fizičnih učinkov (na primer gravitacije),
- sistem za umetno inteligenco,
- omrežno podporo,

- za izdelavo grafičnega uporabniškega vmesnika (GUI) ima vključene zunanje knjižnice,
- itd.

Pogon je napisan v programskem jeziku C++, igro pa pišemo v jeziku Python.

Podpira naslednje platforme:

- Microsoft Windows,
- Linux,
- Mac OS X in
- FreeBSD.

OGRE

OGRE ni igralni pogon, ampak le grafični. Za vse ostale stvari moramo uporabiti zunanje knjižnice (OGRE eksplicitno podpira več znanih skupkov knjižnic za razvoj iger) ali pa sisteme napisati sami [31].

Pogon je napisan v programskem jeziku C++.

Podpira naslednje platforme:

- Linux,
- Microsoft Windows,
- Mac OS X,
- NaCl,
- WinRT,
- Windows Phone 8 in
- Android.

Ena bolj znanih komercialnih iger, ki uporablja OGRE, je Torchlight.

jMonkey

Poleg grafičnega sistema vsebuje še [32]:

- vmesnik za tipkovnico, miško itd.,
- sistem za simuliranje fizičnih učinkov (na primer gravitacije),
- omrežno podporo,
- sistem za izdelavo in upravljanje z grafičnim uporabniškim vmesnikom (GUI),
- podporo za več zaslonov,
- itd.

Pogon je napisan v programskem jeziku Java. Tudi delo z njim poteka preko jezika Java.

Prenosljivost torej ni problem, saj je Java podprta skoraj vsepovsod.

Crystal Space

Crystal Space je osnova za izdelavo 3D-aplikacij. Tipično se uporablja kot igralni pogon, vendar je njen nabor funkcij bolj splošno usmerjen [33].

Ponuja nam:

- 2D- in 3D- grafične sisteme,
- zvočni sistem,
- detekcijo trkov,
- sistem za simuliranje fizičnih učinkov na objekte (na primer gravitacije).

Pogon je napisan v programskem jeziku C++, v katerem pišemo tudi igro. Poleg C++ lahko uporabljamo tudi Python, Perl ali Javo, vendar najbolje sta podprta jezika C++ in Python.

Podpira naslednje platforme:

- Mac OS X,
- Microsoft Windows in
- Linux.

4.3 Brezplačne knjižnice za razvoj iger

Knjižnica je vnaprej napisana koda, ki jo uvozimo in uporabimo za izvršitev posameznih nalog. Na primer: uvoz grafične vsebine, detekcija pritisnjene tipke na tipkovnici itd. V nadaljevanju si bomo ogledali dve znani knjižnici za razvoj računalniških iger - SDL in SFML.

SDL (Simple DirectMedia Layer)

SDL je multimedijska knjižnica s široko podporo različnim platformam. Zagotovi nizkonivojski dostop do zvočnih naprav, tipkovnice, miške, grafičnih kartic itd. Uporabljajo jo video predvajalniki, emulatorji in več popularnih iger [34].

SDL uradno podpira naslednje OS oziroma platforme: Linux, Windows, Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, in QNX.

Neuradno pa so podrte še: AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS in OS/2.

SDL je napisan v programskem jeziku C, vendar deluje nativno tudi z jezikom C++. S knjižnico lahko delamo tudi v mnogih drugih jezikih: Ada, C#, D, Eiffel, Erlang, Euphoria, Go, Guile, Haskell, Java, Lisp, Lua, ML, Oberon/Component Pascal, Objective C, Pascal, Perl, PHP, Pike, Pliant, Python, Ruby, Smalltalk in Tcl.

SDL je izdan pod licenco GNU LGPL, ki dovoljuje brezplačno uporabo za razvoj komercialnih programov in iger, če jih povežemo z dinamično knjižnico.

SFML (Simple And Fast Multimedia Library)

SFML podpira manj platform in programskih jezikov kot SDL, vendar je bližje igralnim pogonom, saj ponuja veliko visokonivojskih funkcij [35].

Sestavljena je iz naslednjih modulov:

- sistemski modul (niti in časovniki),
- modul za okna (upravlja okna in uporabniško interakcijo),
- grafični modul (enostaven prikaz slik in objektov),
- zvočni modul (vmesnik za upravljanje zvočnih učinkov in glasbe) ter

- omrežni modul (delo z vtičnicami na prenosljiv način).

Vsi moduli soodvisni od sistemskega in grafični je odvisen od modula za okna. Poleg teh omejitev jih lahko uprabljamo samostojno.

Uradno so podprti naslednji programski jeziki: C, C++, .NET, Python, Ruby in D.

Neuradno pa lahko uporabimo še: Ocaml ali Java.

Podpira naslednje platforme: Windows, Mac OS X in Linux.

4.4 Brezplačna orodja za razvoj iger

Ta orodja omogočijo izdelavo iger ljudem brez znanja o programiranju. Ponavadi vsebujejo IDE z GUI in preprostim skriptnim jezikom. Tukaj navajam dve izmed teh orodij.

RPG Maker

RPG Maker je program, ki omogoči uporabnikom kreacijo iger igranja vlog.

Vključuje [36]:

- urejevalnik, s katerim izgradimo stopnje,
- preprost skriptni jezik za skriptiranje dogodkov in
- urejevalnik za definiranje bitk.

Program je naložen z veliko vsebine (grafika, zvočni učinki, glasba itd.), ki jo uporabnik lahko uporabi za kreacijo iger. Poleg tega pa lahko uporabniki dodajajo vsebino, ki so jo izdelali sami ali jo prenesli s spleta. Obstaja mnogo spletnih strani, ki imajo na voljo na tisoče brezplačnih vsebin za prenos.

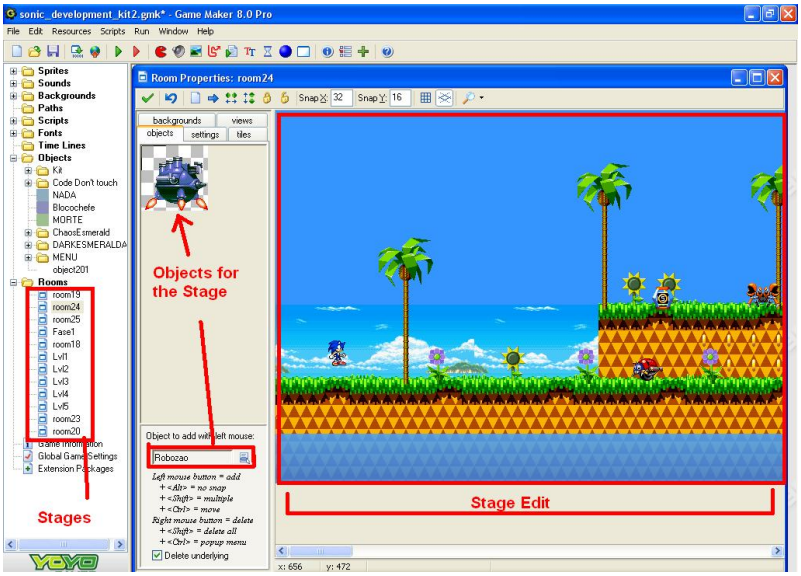
Slika 4.1: Eternal Eden - posnetek zaslona [37], ki nam predstavi tudi tipičen videz vseh iger, izdelanih s programom Rpg Maker

Poleg velikega števila brezplačnih iger, ki so bile izdelane s tem programom, je bilo izdelanih tudi nekaj komercialnih iger (na primer igra Eternal Eden, ki jo lahko vidimo na sliki 4.1).

GameMaker

GameMaker je bil izdelan zato, da bi uporabnikom omogočil enostaven razvoj računalniških iger brez znanja kompleksnih programskih jezikov, kot sta C++ ali Java. Kot lahko vidimo na sliki 4.2, se v GameMaker IDE se večina dela opravi z miško z vlečenjem elementov na ustrezno mesto. Tako lahko uporabniki brez izkušenj s programiranjem kreirajo računalniške igre le z razvrščanjem ikon na zaslonu. Bolj napredni uporabniki lahko uporabijo vključen skriptni jezik za izdelavo bolj zahtevnih iger [38].

Glavna verzija programa podpira delo na platformah Windows in Mac OS X, na katerih bo tudi delovala končana igra. Obstaja pa še veliko drugih verzij, ki omogočijo kreacijo iger za IOS, Android, Windows Phone 8 itd.



Slika 4.2: Game Maker IDE - posnetek zaslona [39]

4.5 MonoXNA

MonoXNA je implementacija ogrodja XNA, ki omogoči razvoj iger za Windows, Mac OS X in Linux z uporabo OpenGL za 3D grafiko. Zadnja verzija je bila izdana leta 2010.

MonoGame je uporabil veliko delov MonoXNA za svojo osnovo in je spiritualni naslednik MonoXNA [40, 41].

4.6 MonoGame

MonoGame je brezplačna implementacija Microsoft XNA 4.0. Cilj te implementacije je omogočiti XNA-programerjem enostaven prenos iger na naslednje platforme [41]:

- iOS,
- Android,
- Mac OS X,
- Linux,

- Playstation Mobile,
- Windows 8,
- Windows 8 Metro,
- Windows Phone 8 in
- OUYA.

4.7 ANX.Framework

ANX.Framework je ogrodje za razvoj računalniških iger, kompatibilno z ogrodjem XNA. Izvorna koda, napisana za Microsoft XNA, deluje brez sprememb z ANX.Framework, zamenjati moramo le imenski prostor (angl. namespace).

ANX.Framework trenutno zamenja DirectX 9, ki ga uporablja ogrodje XNA za svoj grafični, zvočni in vhodni sistem z DirectX 10. V prihodnosti pa namerava podpirati še DirectX 11, DirectX 11.1 in OpenGL 3. OpenGL 3 bo omogočil enostaven prenos iger na platforme, ki jih Microsoft XNA ne podpira [42].

4.8 XNI

XNI je implementacija ogrodja XNA 4.0 v programskem jeziku Objective-C, ustvarjena v namen razvoja iger za iPhone in iPad v znani objektno usmerjeni arhitekturi.

Je statična knjižnica za uporabo z razvojnim orodjem Xcode. Na voljo nam ponudi ožji nabor razredov iz ogrodja XNA. Knjižnica je v zgodnjih začetkih razvoja in podpira samo večje zmogljivosti ogrodja XNA.

Monkey Labour je bila prva izdana igra, ki je uporabila XNI. Poleg nje so študenti izdelali še več krajših iger.

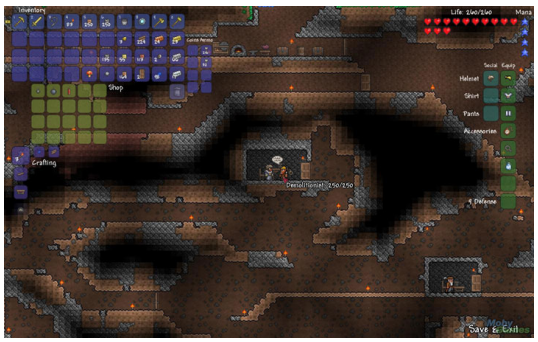
Avtor Matej Jan je osnovo za implementacijo izdelal leta 2010 za projekt, ki je zajemal izdelavo igre za iPhone. Kasneje je pomagal pri zasnovi in izvedbi predmeta na Fakulteti za računalništvo in informatiko v Ljubljani. Predmet je v celoti namenjen razvoju iger (za iOS) [43, 44].

Poglavje 5

Komercialne igre, izdelane z Microsoft XNA

Spisek iger, ki uporabljajo ogrodje XNA, je dolg. Veliko iger tega dejstva tudi nikoli ne omeni, tako, da so na spletu omenjeni le delni spiski iger, za katere se ve, da uporabljajo ogrodje XNA. Tukaj si bomo pogledali le nekaj bolj znanih komercialnih iger. Večina teh iger je tudi tako imenovanih neodvisnih iger, kar nam kaže primernost Microsoft XNA za uporabo s strani samostojnih razvijalcev.

`Terraria` je akcijsko-pustolovska RPG-igra (angl. action-adventure RPG). Igra vsebuje elemente raziskovanja, izdelovanja predmetov in stavb ter bitk s pošastmi v naključno generiranemu 2D-svetu. Igra ima veliko podobnosti z zelo znano igro Minecraft. Kot lahko vidimo na sliki 5.1, je Terraria 2D-igra v nasprotju z Minecraft, ki je 3D-igra.



Slika 5.1: Terraria - posnetek zaslona [45]

Igro je razvilo podjetje Re-Logic, v katerem sta zaposlena le dva posa-

meznika - programer Andrew Spinks in grafični oblikovalec Finn Bryce. Za glasbo je poskrbel zunanji kontraktor Scott Lloyd Shelley.

Možno jo je bilo kupiti preko spletne trgovine Steam. V prvem dnevu je bilo prodanih več kot 50 tisoč kopij, in v prvem tednu pa 200 tisoč kopij. Bila je najbolj prodajana igra v spletni trgovini Steam tisti teden in je premagala komercialne velikane, kot sta Witcher 2 in Portal 2 [46].

`Magicka` je akcijsko-pustolovska igra. Štirje čarovniki se bojujejo proti zlobnemu čarovniku in pošastim, ki jih je kreiral. Igra je zastavljena zelo neresno, z veliko humorja in nestandardnimi orožji za fantazijsko zvrst. Posnetek zaslona iz igre lahko vidimo na sliki 5.2.

Igro je razvilo osem študentov švedske univerze za tehnologijo.

Do januarja 2012 je bilo prodanih več kot milijon izvodov [47].

Slika 5.2: Magicka - posnetek zaslona [48]

`Bastion` je akcijska RPG. Svet, v katerem je zastavljena igra, je sestavljen iz lebdečih otočkov (na sliki 5.3 vidimo en otoček in ozadje) in ko se igralec približa robu, se mu prikažejo poti do naslednjega otočka. V fantazijskem svetu je veliko pošasti, ki hočejo igralca poškodovati. Igralec ima na voljo več orožij.

Igro je izdelala ekipa sedmih ljudi.

V letu 2011 je bilo prodanih več kot 500 tisoč kopij [49].

Slika 5.3: Bastion - posnetek zaslona [50]

`AI War: Fleet Command` je realno-časovna strategija. Igralec nabira resurse in gradi vesoljske ladje (te lahko vidimo na sliki 5.4), ki jih nato uporabi za bojevanje z AI-nasprotniki, ki so močnejši od njega. Agresivnost in težavnost nasprotnikov je odvisna od igralčevih akcij v sami igri.

Igro je razvila ekipa sedmih ljudi.

Glede na informacije, ki jih je objavil glavni razvijalec Christopher M. Park, je bilo prodanih od 20 do 30 tisoč kopij [51].

Slika 5.4: AI War: Fleet Command - posnetek zaslona [52]

Z ogrodjem XNA so torej neodvisni razvijalci izdelali mnogo uspešnih komercialnih iger v različnih žanrih.

Poglavje 6

Kako nam ogrodje XNA pomaga pri razvoju iger

Ogledali si bomo, kako nam ogrodje XNA pomaga pri razvoju iger na splošno in katere funkcionalnosti tega ogrodja sem uporabil pri izdelavi demonstracijske igre. To poglavje nikakor ne zajema celotne funkcionalnosti, ki nam jo ponuja ogrodje XNA.

6.1 Imenski prostor Microsoft.XNA.Framework

Osnovni imenski prostor nam poda razrede in strukture oziroma podatkovne tipe, ki jih pogosto potrebujemo. Podrobnosti o spodaj naštetih razredih in podatkovnih tipih si lahko pogledamo v delnih razrednih diagramih v dodatku A (strani A.1 in A.2).

Oglejmo si nekaj od teh razredov in podatkovnih tipov:

- Razred `Game` nam poda osnove za grafično inicializacijo, igrino logiko, izrisovalno kodo in igrino zanko. Več o strukturi, ki nam jo poda ta razred, smo si že ogledali v poglavju 3.1.4 Enostavna in razumljiva osnovna struktura izvorne kode.

- Razreda `GameComponent` in `DrawableGameComponent` - razred `GameComponent` nam ponuja modularen način dodajanja funkcionalnosti igri. Če komponenta naloži in riše grafično vsebino igre, uporabimo razred `DrawableGameComponent`. Igrino logiko oziroma izrisovalno kodo v komponento dodamo na enak način kot v razredu `Game`. Komponento

registriramo tako, da jo podamo metodi `Game.Components.Add`. Ko je komponenta registrirana, bodo njene metode `Update`, `Initialize` in `Draw` klicane iz `Game.Update`, `Game.Initialize` in `Game.Draw`. Za vsako komponento lahko določamo vrstni red risanja, vidnost (se kliče metoda `Draw` ali ne) in aktivnost (se kliče metoda `Update` ali ne).
V demonstracijski igri so komponente uporabljene za ločitev igre na logične neodvisne dele. V osnovi je igra razdeljena na komponente za umetno inteligenco, izrisovanje, fiziko, zvok, upravljanje s stanji in generiranje stopenj. Nekatere od teh komponent vsebujejo še dodatne podkomponente. Sistem za animacijo je na primer podkomponenta izrisovalne komponente.

- Razred `GraphicsDeviceManager` nam omogoča nastavitev in spreminjanje osnovnih grafičnih nastavitev. Z njegovo pomočjo lahko spreminjamo ločljivost zaslona, določimo, ali igra teče v oknu ali na celem zaslonu, ter spreminjamo še nekatere naprednejše nastavitve (na primer intenzivnost glajenja robov, globinske šablone itd.). Za spreminjanje grafičnih nastavitev je ta razred uporabljen tudi v demonstracijski igri.

```
// Vzpostavi nove nastavitve
public static void ApplyGraphicSettings()
{
  graphics.PreferredBackBufferWidth = (int)Constants.viewport.X;
  graphics.PreferredBackBufferHeight = (int)Constants.viewport.Y;
  graphics.IsFullScreen = Constants.fullScreen;
  graphics.ApplyChanges();
}
```

- Razred `GameTime` nam vrne v več oblikah čas, ki je pretekel od zadnjega klica metode `Game.Update`. S to informacijo lahko zagotovimo, da vse v igri teče s pravilno hitrostjo. V demonstracijski igri je razred `GameTime` uporabljen še za zagotovitev enakomernih časovnih razmikov med vali sovražnikov oziroma stopnjami in prikazi tekstovnih sporočil igralcu za določen čas.

```
public override void Update(GameTime gameTime)
{
  // čas v sekundah od zadnjega klica metode Update
  float elapsed = (float)gameTime.ElapsedGameTime.TotalSeconds;
  ...
  ...
}
```

- Razred `MathHelper` vsebuje metode, ki nam poenostavijo nekatere pogoste matematične operacije, in definira uporabne konstante (na primer Pi). Metoda `Clamp` je uporabljena v demonstracijski igri, kadar morajo vrednosti spremenljivk ostati v vnaprej določenem območju.

```
// Preveri če je vrednost med min in maks in jo ustrezno popravi če ni
vrednost = MathHelper.Clamp(vrednost, min, maks);
// Uporabne definirane konstante
pi = MathHelper.Pi;
```

- Podatkovni tipi `BoundingBox`, `BoundingSphere`, `Ray`, `Plane` nam poenostavijo detekcijo trkov v 3D-igrah.

- Podatkovni tipi `Rectangle` in `Point` nam poenostavijo detekcijo trkov v 2D-igrah. Demonstracijska igra uporablja podatkovni tip `Rectangle` za detekcijo trkov med igralcem, sovražniki, trofejami in strelivom.

```
// vhod: 2 x prostor, ki ga zaseda 2D tekstura na zaslonu
// izhod: true - se prekrivata, false - se ne prekrivata
public static Boolean Check(Rectangle a, Rectangle b)
{
  if (a.Intersects(b))
    return true;
  else
    return false;
}
```

- Podatkovni tipi `Vector2`, `Vector3`, `Vector4` shranijo dve, tri oziroma štiri vrednosti tipa `Float`. Uporabljamo jih za shranjevanje smeri in magnitude ter tudi koordinat v 2D- oziroma 3D-prostoru. Podajo nam tudi metode za seštevanje, množenje, normalizacijo vektorjev in transformacijo vektorjev z uporabo matrik. Kot lahko vidimo v naslednjem primeru, demonstracijska igra uporablja podatkovni tip `Vector2` za shranjevanje koordinat in normaliziranih smernih vektorjev, hkrati pa tudi uporabi funkcionalnost odštevanja dveh vektorjev in normalizacije vektorja.

```
// vhod: trenutna pozicija na zaslonu,
//       ciljana pozicija na zaslonu
// izhod: normaliziran smerni vektor
private Vector2 GetDirection(Vector2 origin, Vector2 target)
{
  Vector2 result;
  result = target - origin;
  result.Normalize();
```

```
  return result;
}
```

- Podatkovni tip `Matrix` uporabljamo v 3D-igrah za definicijo in delo s kamerami ter shranjevanje trenutne postavitve 3D-objekta v svetu. Poda nam veliko metod za delo z matrikami - rotacije, translacije itd. Pri delu s kamerami nam zelo pomaga metoda `Matrix.CreateLookAt`.

- Podatkovni tip `Color` - enostavna representacija barve, ki je lahko izbrana izmed mnogih prednastavitev ali definirana v RGB formatu. Demonstracijska igra uporabi podatkovni tip `Color` pri izrisovanju na zaslon in pri nalagalniku za izgradnjo prikazovalnika napredka.

```
Color color;
// prednastavljena barva
color = Color.Red;
// nastavitev RGB barve
color = new Color(255,0,0);
// nastavitev RGB barve z alfa
color = new Color(255,0,0,100);
```

6.2 Imenski prostor Microsoft.XNA.Framework.Graphics

Imenski prostor `Microsoft.XNA.Framework.Graphics` vsebuje nizko nivojski API, ki uporablja strojno pospeševanje za prikaz 3D-objektov. V dodatku A na straneh A.3 in A.4 najdemo tudi delni razredni diagram tega imenskega prostora, ki vsebuje podrobnosti za večino razredov in podatkovnih tipov, opisanih v nadaljevanju.

Oglejmo si nekaj razredov oziroma podatkovnih tipov, ki nam jih poda ta imenski prostor:

- Podatkovni tip `Texture2D` - v podatkovni tip `Texture2D` naložimo in hranimo vso grafično vsebino v 2D-igri. Omogoča nam pridobivanje informacij o velikosti slike in barvi posameznih pikslov. V 2D-igrah lahko informacijo o alfa vrednosti posameznih pikslov uporabimo pri implementaciji metode za detekcijo trkov med objekti z natančnostjo do enega piksla.

- Razred `GraphicsDeviceCapabilities` smo uporabljali za preverjanje zmogljivosti grafične kartice v starejših verzijah ogrodja XNA. V ogrodju XNA 4.0 je bilo to poenostavljeno z izbiro med dvema profiloma v projektnih nastavitvah pred prevajanjem in izgradnjo projekta.

- Razred `PresentationParameters` vsebuje grafične parametre, ki smo jih spreminjali z uporabo razreda `GraphicsDeviceManager`.

- Razred `GraphicsDevice`, ki nam omogoča direkten dostop do predpomnilnika grafične kartice, risanje trikotnikov, senčenje pikslov ali trikotnikov itd.

- Podatkovni tip `SpriteFont` nam hrani pisave in pomaga pri delu z njimi. Metoda `SpriteFont.MeasureString()`, ki nam vrne dolžino teksta, napisanega s to pisavo v pikslih, je v veliko pomoč pri postavitvi teksta na zaslon na pravilnem mestu.

```
// Izmerimo dimenzije teksta prikazanega s to pisavo
Vector2 titleOrigin = font[0].MeasureString(screen.menuTitle) / 2;
// Vpliv velikosti pisave
titleOrigin.X *= titleScale;
titleOrigin.Y *= titleScale;
// To informacijo uporabimo pri risanju teksta na sredino zaslona
spriteBatch.DrawString(font[0], screen.menuTitle, titlePosition, titleColor, 0,
    titleOrigin, titleScale, SpriteEffects.None, 0);
```

- Z razredom `Spritebatch` izrisujemo na zaslon podatkovna tipa `Texture2D` in `Spritefont`. `Spritebatch` omogoča veliko načinov izrisovanja 2D-grafike.

```
...
// risanje s privzetimi nastavitvami
spriteBatch.Begin();
// risanje tekstura
spriteBatch.Draw(tekstura, pozicija, barva);
// risanje teksta
spriteBatch.DrawString(pisava, tekst, pozicija, barva);
spriteBatch.End();
// nastavitve oz. opcije pri risanju s spritebatch
spriteBatch.Begin(
        SpriteSortMode.BackToFront,
        BlendState.Additive,
        SamplerState.AnisotropicClamp,
        DepthStencilState.Default,
```

```
            RasterizerState.CullClockwise,
            efekt,
            transformacijskaMatrika);
...
```

- Razred `Model` - predstavlja 3D-model in vsebuje vse metode, potrebne za delo z modeli, njihovimi deli in risanje modelov na zaslon.
- Razred `Effect` - omogoča uporabo HLSL za senčenje in tudi uporabo raznovrstnih drugih učinkov ter tehnik pri risanju tekstur in modelov.
- Struktura `Viewport` - uporabna pri risanju, saj vedno hrani dimenzije in razmerje zaslona.

6.3 Imenski prostor Microsoft.XNA.Framework.Content

Vsebuje izvajalne komponente od cevovoda vsebine. Razredni diagram za ta imenski prostor si lahko ogledamo v dodatku A na strani A.5.

Razred `ContentManager` uporabimo med izvajanjem igre za nalaganje vsebine iz binarnih datotek, ki so bile izdelane med prevajanjem oziroma izgradnjo projekta. Upravlja tudi z življenjsko dobo vseh naloženih objektov.

V demonstracijski igri uporabljamo ta imenski prostor za nalaganjem vseh delov vsebine, ki niso razširljivi (vnaprej vemo točno, koliko jih je); na primer ozadja v menijih, teksture za GUI itd. Razširljivi deli vsebine (teksture za ladje, animacije, zvočni efekti itd.) so naloženi direktno brez uporabe cevovoda vsebine, saj tako omogočimo uporabnikom dodajanje poljubnega števila novih delov vsebine brez dostopa do izvorne kode in njenega ponovnega prevanja/grajenja.

```
Texture2D bar;
// nalaganje teksture iz cevovoda vsebine
bar = Content.Load<Texture2D>(@"Teksture\GUI\bar");
```

6.4 Imenski prostor Microsoft.XNA.Framework.Input

Imenski prostor `Microsoft.XNA.Framework.Input` vsebuje razrede, ki nam omogočajo dostop do tipkovnice, miške in kontrolnih pripomočkov Xbox 360.

Podrobnosti si lahko ogledamo v razrednem diagramu tega imenskega prostora, ki ga najdemo v dodatku A na strani A.7. V imenskem podprostoru `Input.Touch` najdemo še razrede, ki nam omogočijo delo z napravami z upravljanjem na dotik.

V demonstracijski igri uporabljamo ta imenski prostor za kontrolo igralčeve ladje in za delo z meniji.

```
// pridobimo trenutno stanje tipkovnice
KeyboardState keyboardState = Keyboard.GetState();
// preverimo, stanje posameznih tipk
if(keyboardState.IsKeyDown(Keys.Space))
  //ukrepamo
```

6.5 Imenski prostor Microsoft.XNA.Framework.Audio

Imenski prostor `Microsoft.XNA.Framework.Audio` vsebuje nizkonivojske API-metode, ki lahko naložijo in uporabijo XACT-projekte ter zvočne datoteke iz cevovoda vsebine za predvajanje zvoka.

Demonstracijska igra naloži zvočne učinke v podatkovni tip `SoundEffect` ter ustvari `SoundEffectInstance,` s katerim upravljano naložene zvočne učinke. Razred `SoundEffectInstance` vsebuje metode za upravljanje s stanjem predvajanja (ustavi, nadaljuj, pavziraj, itd.) in nastavitev glasnosti. Razredni diagram v dodatku A na strani A.6 vsebuje podrobnejše informacije o teh podatkovnih tipih in drugih komponentah tega imenskega prostora.

```
// ustvari novo instanco
soundEffectInstance = soundEffect.CreateInstance();
soundEffectInstance.Play(); // predvajaj
soundEffectInstance.Volume = 0.5f; // 50% glasnost
```

6.6 Imenski prostor Microsoft.XNA.Framework.Media

Imenski prostor `Microsoft.Xna.Framework.Media` vsebuje razrede, ki nam omogočijo enumeracijo, predvajanje in pregled pesmi, albumov, slik.

S pomočjo razreda `MediaPlayer` v demonstracijski igri predvajamo glasbo. Metode v tem razredu nam omogočijo upravljanje s stanjem predva-

janja (ustavi, nadaljuj, pavziraj itd.) in nastavitev glasnosti. Za shranjevanje pesmi uporabimo podatkovni tip `Song`.

```
MediaPlayer.Play(song); // predvajaj pesem
MediaPlayer.Pause(); // pavziraj predvajanje
MediaPlayer.Volume = 0.5f; // 50% glasnost
```

6.7 Imenski prostor Microsoft.XNA.Framework.Net

Imenski prostor `Microsoft.Xna.Framework.Net` vsebuje razrede, ki implementirajo podporo za Xbox Live oziroma Games For Windows Live.

V demonstracijski igri tega imenskega prostora ne uporabimo, saj igra ni večigralska in tudi nimamo Microsoftovega dovoljenja za uporabo njihovih spletnih storitev.

6.8 Imenski prostor Microsoft.XNA.Framework.Storage

Imenski prostor `Microsoft.Xna.Framework.Storage` vsebuje razrede, ki omogočajo branje in pisanje datotek. Uporabimo ga lahko na primer za shranjevanje in nalaganje stanja igre.

Demonstracijska igra je zelo preprosta in ne potrebuje teh metod. Vse, kar je treba shraniti, so trenutna stopnja in točke, ki jih je zaslužil igralec. Pri funkciji avtomatskega shranjevanja igre na začetku stopnje in nalaganja shranjenih iger uporabimo standardne vhodno-izhodne funkcije programskega jezika C#.

6.9 Imenski prostor Microsoft.XNA.Framework.GamerServices

Imenski prostor `Microsoft.Xna.Framework.GamerServices` vsebuje razrede, ki implemetirajo razne spletne storitve, ki so direktno povezane z Xbox Live oziroma Games For Windows Live, na primer interakcijo s profili igralcev, dosežki itd.

V demonstracijski igri tega imenskega prostora ne uporabimo, saj igra ni večigralska in tudi nimamo Microsoftovega dovoljenja za uporabo njihovih spletnih storitev.

Poglavje 7

Izdelava demonstracijske igre

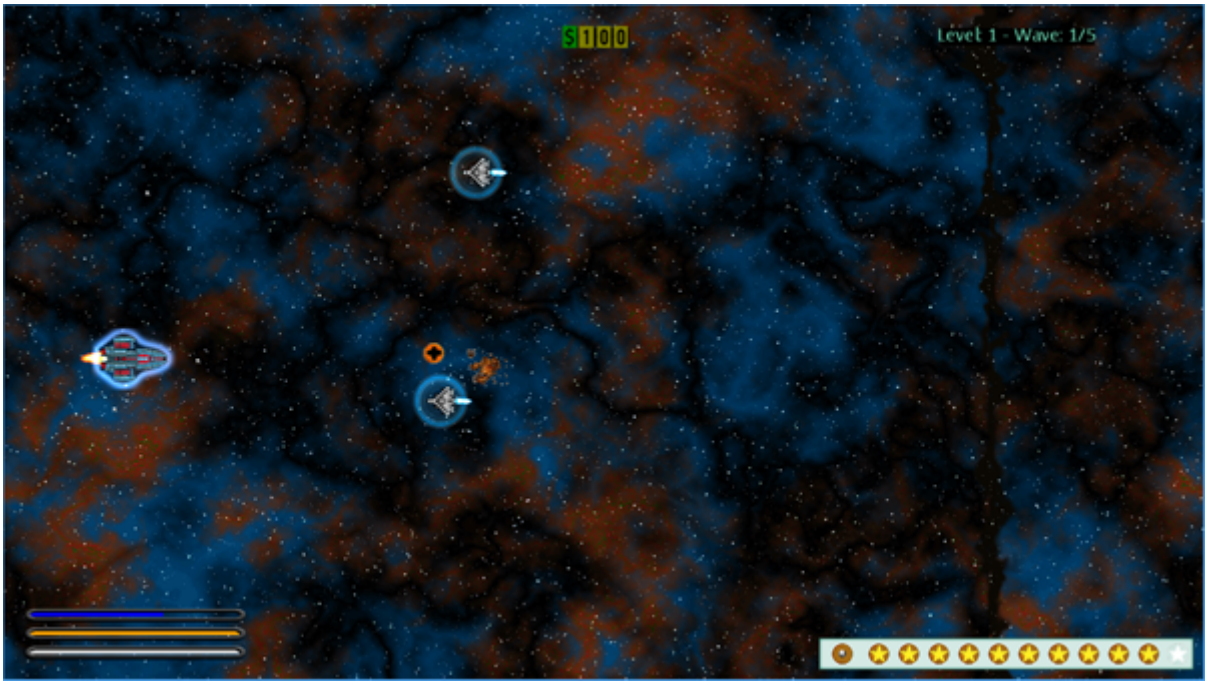
V tem poglavju si bomo pogledali splošen opis demonstracijske igre in nekatere podrobnosti njene izdelave, ki niso direktno povezane z ogrodjem XNA.

7.1 Splošen opis

Izdelal sem 2D-vesoljsko streljačino z avtomatskim premikanjem zaslona v desno stran (angl. "side-scrolling 2D space shooter").

V igri igralec kontrolira vesoljsko ladjo (lahko jo premika navzgor in navzdol po zaslonu ter strelja). Igralčeva ladja leti v desno stran (igralec jo lahko tudi premika v desno in levo stran, vendar le znotraj zaslona). Sovražniki priletijo z desne strani in potujejo proti levi strani zaslona mimo igralčeve ladje. Dokler so na zaslonu, poskušajo sestreliti igralčevo ladjo z raznimi strelivi. Igralec zasluži točke, ko uniči sovražne vesoljske ladje. Na koncu stopnje mora igralec premagati večjega sovražnika - šefa. Ko ga premaga, je stopnja uspešno končana in začne se nova stopnja. Ob začetku nove stopnje se igra avtomatsko shrani.

Na sliki 7.1 lahko vidimo videz demonstracijske igre. V spodnjem levem kotu je prikazovalnik trenutnega stanja igralčeve ladje (ščit, oklep, struktura). V desnem spodjem kotu vidimo, katero orožje uporablja igralčeva ladja in na katero stopnjo je bilo orožje nadgrajeno. Orožje višje stopnje strelja bolj hitro in močneje. Igralec nadgradi orožje s pobiranjem zvezd, ki včasih nastanejo ob uničenju sovražne ladje. V desnem zgornjem kotu vidimo trenutno stopnjo in napredek igralca po tej. Na sredini zgoraj so prikazane igralčeve točke.



Slika 7.1: Igra - posnetek zaslona

7.2 Modularnost

Igra je razdeljena na logične komponente, prikazane na sliki 7.2. Vsaka komponenta opravlja svojo funkcijo in je ne zanima, kaj počnejo ostale komponente. Tukaj si bomo ogledali, kaj delajo posamezne komponente.

7.2.1 Nalagalnik

Nalagalnikova naloga je naložiti vso vsebino igre (slike, zvočne učinke, pisave, glasbo itd.) in jih dati na razpolago komponentam, ki jih potrebujejo.

Demonstracijska igra omogoča uporabniku modifikacijo obstoječe vsebine in dodajanje poljubne količine nove vsebine brez dostopa do izvorne kode oziroma znanja programiranja. To je realizirano z uporabo XML-datotek, ki definirajo posamezne entitete in njim pripadajočo vsebino ter jih povežejo z drugimi entitetami in vsebino preko imena. Poglejmo si eno tako XML-datoteko s katero je definirana sovražna ladja:

```
<?xml version="1.0" encoding="utf-8" ?>
<ship>
  <name>fighter001</name>
```

Slika 7.2: Moduli - diagram

```
  <type>fighter</type>
  <bullet>smallGauss</bullet>
  <bulletlevel>3</bulletlevel>
  <explosionanimation>fighterExplosion</explosionanimation>
  <speed>80</speed>
  <hull>100</hull>
  <armor>100</armor>
  <shield>100</shield>
  <ai>basic</ai>
  <exhaustanimation>fighterExhaust</exhaustanimation>
  <isanimated>false</isanimated>
  <animation>null</animation>
</ship>
```

Nalagalnik prebere to XML-datoteko in takoj naloži še sliko z istim imenom, ki predstavlja to sovražnikovo ladjo. Vidimo tudi nekaj povezav z drugo definirano vsebino igre:

- (bullet)smallGauss(/bullet) je strelivo, ki je definiramo s svojo XML-datoteko, kjer je določeno tudi ime, ki ga uporabimo tukaj,
- (explosionanimation)fighterExplosion(/explosionanimation) je animacija, ki se predvaja, ko je sovražnik uničen, in je ravno tako definirana s svojo XML-datoteko.

Z XML-datotekami so definirani sovražniki, strelivo, trofeje, glasba, zvoki,

animacije, igralčeva ladja in stopnje. Vse te entitete in vsebino lahko torej uporabnik poljubno spreminja, dodaja, odstranjuje in povezuje v celoto.

7.2.2 Scena

Scena hrani vse trenutno aktivne entitete in strani. Do teh entitet in strani dostopa večina ostalih komponent in nad njimi izvaja razne operacije.

Scena je popolnoma pasivna komponenta in le hrani podatke.

7.2.3 Komponenta za generiranje stopenj

Komponenta za generiranje stopenj prebere parametre za stopnjo, ki jih je naložil nalagalnik, in potem glede na te parametre generira vse potrebne sovražnike ter jih doda na sceno.

Parametre si lahko ogledamo na spodnjem primeru:

```
<?xml version="1.0" encoding="utf-8" ?>
<level>
  <name>level1</name>
  <waves>5</waves>
  <allowedships>
    <ship>
      <shipname>fighter001</shipname>
      <composition>0.80</composition>
    </ship>
    <ship>
      <shipname>bomber001</shipname>
      <composition>0.20</composition>
    </ship>
  </allowedships>
  <boss>boss001</boss>
</level>
```

Parametri torej zajemajo dolžino stopnje (število valov sovražnikov), imena vseh sovražnikov ter njihovo pogostost in ime sovražnika, ki ga uporabimo za šefa.

7.2.4 Komponenta za upravljanje s stanjem igre

Komponenta za upravljanje s stanjem igre upravlja aktivne strani na sceni in poskrbi za njihovo dodajanje ter odstranjevanje. V domeni te komponente

je tudi nadzor (tipkovnica, miška).

Primeri strani: ozadje v menijih, meniji, grafični uporabniški vmesnik, igra (stran, ki vsebuje nadzor igračeve ladje in pogojuje izris entitet), nalagalnik itd.

Strani lahko poljubno sestavljamo skupaj, tako so na primer med igranjem igre aktivne tri strani - premikajoče ozadje, igra in grafični uporabniški vmesnik. V primeru, da igro pavziramo, se na sceno doda še četrta stran, ki vsebuje meni za izhod/nadaljevanje igre.

7.2.5 Izris

Izris je samostojna komponenta, sestavljena iz mnogih podkomponent. V tej komponenti lahko popolnoma spremenimo način prikaza igre na zaslonu (na primer iz 2D-prikaza v 3D-prikaz), saj komponenta le prebere podatke o entiteti oziroma strani na sceni in jih potem izriše na poljuben način. Uporaba podkomponent za izris posameznih delov igre na zaslon nam omogoča enostavno spreminjanje vidnosti, posodabljanja in vrstnega reda risanja teh delov.

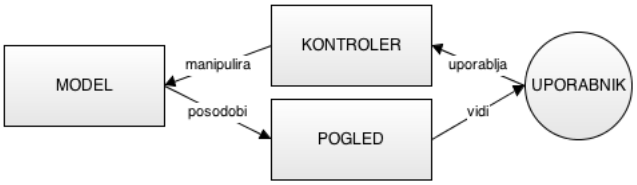
```
gameplayScreenRenderer.DrawOrder = 5;
guiScreenRenderer.DrawOrder = 6;
animationSystem.Enabled = false;
guiScreenRenderer.Visible = false;
```

Na zgornjem primeru lahko vidimo, kako se določi vrstni red risanja komponent. V našem primeru smo določili, da se igra izriše pred grafičnim uporabniškim vmesnikom.

V nadaljevanju smo izključili posodabljanje sistema za animacije. Animacije se še vedno rišejo na zaslon, vendar brez klica metode Update, in rezultat so zamrznjene animacije.

Na koncu še izključimo risanje grafičnega uporabniškega vmesnika. Vmesnik se torej še naprej posodablja, vendar ga ne rišemo na zaslon.

Uporabljen je princip programske arhitekture MVC, ki določa, da se prikaz informacij loči od njihovega spreminjanja [50]. Tipično MVC-zasnovo lahko vidimo na sliki 7.3.



Slika 7.3: Tipične komponente MVC [53]

7.2.6 Komponenta za umetno inteligenco

Komponenta za umetno inteligenco poskrbi za naključne in načrtne spremembe smeri potovanja entitet in da sovražniki streljajo na igralca. Obnašanje sovražnikov je definirano v XML-datoteki, ki jo naloži nalagalnik. Parameter (ai)zikzak(/ai) na primer povzroči obračanje vertikalne komponente smernega vektorja vsake tri sekunde. To si lahko ogledamo na spodnjem primeru.

```
else if (ship.Ai == "zikzak")
{
  ship.TimeSinceLastCourseChange += elapsed;
  if (ship.TimeSinceLastCourseChange > 3.0f)
  {
    ship.DirectionY *= -1;
    ship.TimeSinceLastCourseChange = 0f;
  }
}
```

7.2.7 Komponenta za fiziko

Komponenta za fiziko prebere smerni vektor in hitrost za vsako komponento na sceni ter jo premakne na ustrezno mesto glede na pretečen čas. Preveri tudi, ali je prišlo do trka med dvema entitetama in o tem dogodku obvesti entiteto, ki se zna sama ustrezno odzvati na trk z različnimi vrstami entitet.

Na spodnjem primeru vidimo, kako ta komponenta izračuna novo pozicijo entitete z uporabo smernega vektorja, pretečenega časa in hitrosti potovanja entitete.

```
entity.Position += entity.Direction *
    (float)gameTime.ElapsedGameTime.TotalSeconds *
```

entity.Speed;

7.2.8 Komponenta za zvok

Komponenta za zvok je razdeljena na podkomponento za glasbo in podkomponento za zvočne učinke. Podkomponenta za glasbo poskrbi, da se vedno, ko je to ustrezno, predvaja glasba. Podkomponenta za zvočne učinke pa sprejema zahteve za predvajanje zvočnih učinkov in jih izvršuje. Nov zvočni učinek je dogodek, za katerega poslušalec je ta komponenta.

7.2.9 Komponenta za igralnost

Komponenta za igralnost definira in preverja pravila igre. Poskrbi na primer za odstranjevanje vseh entitet, ki niso več na zaslonu. Preveri tudi, ali je bila igralčeva ladja uničena in se mora igra končati.

Služi tudi kot glavna komponenta, saj so v njej definirane ostale komponente. Te komponente se vključijo in izključijo v odvisnosti od trenutnega stanja igra (pavzirana, ustavljena, teče).

7.3 Težave pri izdelavi in končni rezultat

Pri izdelavi igre nisem imel večjih težav, saj sem z Microsoft XNA in Microsoft Visual Studio delal že prej.

Težave logične narave sem odkril hitro s pomočjo razhroščevalnika (angl. debugger), ki je vsebovan v programu Visual Studio. Funkcionalnost vstavljanja prekinitvenih točk (angl. breakpoints) mi je bila pri tem v veliko pomoč. Prekinitveno točko lahko vstavimo kjerkoli v izvorni kodi in ko izvajanje igre pride do naše vstavljene točke, se ustavi in nam omogoči ogled vseh vrednosti naših spremeljivk - izvajanje igre potem lahko nadaljujemo od prekinitvene točke dalje. Z vstavitvijo več prekinitvenih točk lahko torej spremljamo, kako se spreminjajo vrednosti spremeljivk med izvajanjem igre in tako lažje najdemo logične napake v izvorni kodi.

Končni rezultat je igra, zgrajena iz samostojnih modulov.Vsak posamezni sistem oziroma modul bi zlahka uporabil za osnovo pri izdelavi drugih iger. Večina podatkov je v XML-datotekah in tako omogoča tudi zelo enostavno in ekstenzivno možnost modifikacije igre le z uporabo najbolj osnovnega tekstovnega urejevalnika. Vsebina igre (slike, glasba, zvočni učinki, animacije itd.) ni pakirana ali kodirana in jo lahko uporabnik poljubno razširja, ureja

ali odstranjuje njene posamezne dele. Uporabnik lahko torej popolnoma spremeni vse grafične in podatkovne aspekte igre, vendar ne more spremeniti pravil igre. Da bi lahko spremenil še pravila igre, bi morali implementirati skriptni jezik ali veliko prednastavljenih možnosti ter mu jih dati na izbiro.

Poglavje 8

Zaključek

V diplomski nalogi smo si podrobneje ogledali ogrodje XNA in kako nam pomaga pri razvoju iger. Poleg tega ogrodja smo si ogledali še veliko brezplačnih alternativ, ki jih lahko neodvisni razvijalci uporabijo za hiter razvoj iger. Izvedeli smo tudi, da Microsoft ne namerava več podpirati ogrodja XNA oziroma izdati novih različic ogrodja. Kljub temu njegovim uporabnikom ni treba skrbeti, saj jim brezplačne alternative (monoGame, ANX.Framework itd.) omogočajo pisanje izvorne kode v znanem stilu. Te alternative dodajo novo funkcionalnost in podporo za več platform ter eliminirajo nekatere druge slabosti ogrodja XNA (večigralska podpora le preko Microsoft Games For Windows Live oziroma Xbox Live, upravljana koda itd.).

Neodvisni razvijalci so v zadnjih letih z izdajo nekaj zelo znanih iger (na primer Minecraft) vzbudili veliko zanimanja za neodvisne igre. Število neodvisnih iger je naraslo tudi zaradi inovativnih načinov za pridobitev financiranja za razvoj iger (angl. crowdfunding) in novih načinov za razpečevanje iger (na primer spletna trgovina Steam). KickStarter[1] in podobne spletne strani omogočajo neodvisnim razvijalcem, da vnesejo informacije o svoji načrtovani igri v njihovo bazo. Obiskovalci spletne strani se potem odločijo, ali jim je igra všeč, in lahko prispevajo poljubno vsoto denarja ter tako podprejo razvoj igre. V zameno za njihovo podporo dobijo dostop do alfa in beta različič igre ter razne druge dodatke, odvisne od velikosti njihovega prispevka, in tudi končno različico igre, ko je ta izdana. Pri prodaji iger je spletna trgovina Steam uvedla podobno metodo (Steam Greenlight), saj od konca leta 2012 njeni uporabniki z glasovanjem odločajo, katere igre bodo dodane v spletno trgovino. To omogoči kakovostnim neodvisnim igram, da so na željo uporabnikov dodane v spletno trgovino. Spletna trgovina Steam

[1] http://www.kickstarter.com

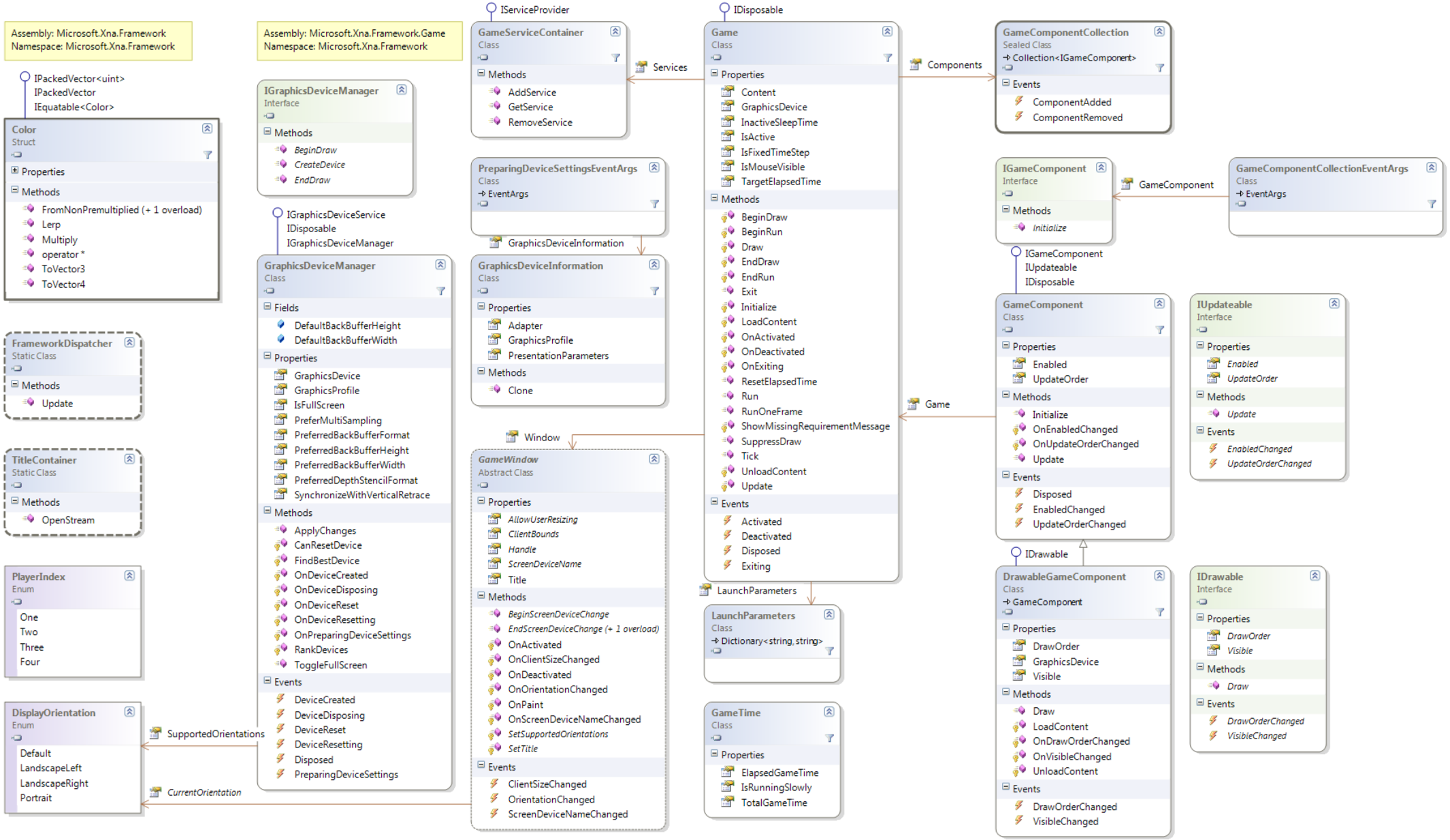
ima po nekaterih podatkih od 50- do 70-odstotni tržni delež pri prodaji iger v digitalni obliki preko spleta [54].

Kombinacija brezplačnih oziroma cenejših orodij za hiter razvoj video iger, ogromen uspeh posameznih neodvisnih iger, novi načini pridobivanja financiranja in razpečave preko spleta ter tudi novi hitro rastoči trgi za neodvisne igre (pametni telefoni, tablični računalniki itd.) so povzročili porast števila neodvisnih iger in neodvisnih razvijalcev.

Dodatek A

Razredni diagrami ogrodja XNA

Razredni diagrami za imenske prostore, ki smo jih veliko uporabljali v demonstracijski igri [55].


Assembly: Microsoft.Xna.Framework
Namespace: Microsoft.Xna.Framework
Assembly: Microsoft.Xna.Framework.Game
Namespace: Microsoft.Xna.Framework
IPackedVector<uint>
IPackedVector
IEquatable<Color>
Color
Struct
Properties
Methods
FromNonPremultiplied (+ 1 overload)
Lerp
Multiply
operator *
ToVector3
ToVector4
FrameworkDispatcher
Static Class
Methods
Update
TitleContainer
Static Class
Methods
OpenStream
PlayerIndex
Enum
One
Two
Three
Four
DisplayOrientation
Enum
Default
LandscapeLeft
LandscapeRight
Portrait
SupportedOrientations
CurrentOrientation
IGraphicsDeviceManager
Interface
Methods
BeginDraw
CreateDevice
EndDraw
IGraphicsDeviceService
IDisposable
IGraphicsDeviceManager
GraphicsDeviceManager
Class
Fields
DefaultBackBufferHeight
DefaultBackBufferWidth
Properties
GraphicsDevice
GraphicsProfile
IsFullScreen
PreferMultiSampling
PreferredBackBufferFormat
PreferredBackBufferHeight
PreferredBackBufferWidth
PreferredDepthStencilFormat
SynchronizeWithVerticalRetrace
Methods
ApplyChanges
CanResetDevice
FindBestDevice
OnDeviceCreated
OnDeviceDisposing
OnDeviceReset
OnDeviceResetting
OnPreparingDeviceSettings
RankDevices
ToggleFullScreen
Events
DeviceCreated
DeviceDisposing
DeviceReset
DeviceResetting
Disposed
PreparingDeviceSettings
IServiceProvider
GameServiceContainer
Class
Methods
AddService
GetService
RemoveService
Services
PreparingDeviceSettingsEventArgs
Class
EventArgs
GraphicsDeviceInformation
GraphicsDeviceInformation
Class
Properties
Adapter
GraphicsProfile
PresentationParameters
Methods
Clone
Window
GameWindow
Abstract Class
Properties
AllowUserResizing
ClientBounds
Handle
ScreenDeviceName
Title
Methods
BeginScreenDeviceChange
EndScreenDeviceChange (+ 1 overload)
OnActivated
OnClientSizeChanged
OnDeactivated
OnOrientationChanged
OnPaint
OnScreenDeviceNameChanged
SetSupportedOrientations
SetTitle
Events
ClientSizeChanged
OrientationChanged
ScreenDeviceNameChanged
IDisposable
Game
Class
Properties
Content
GraphicsDevice
InactiveSleepTime
IsActive
IsFixedTimeStep
IsMouseVisible
TargetElapsedTime
Methods
BeginDraw
BeginRun
Draw
EndDraw
EndRun
Exit
Initialize
LoadContent
OnActivated
OnDeactivated
OnExiting
ResetElapsedTime
Run
RunOneFrame
ShowMissingRequirementMessage
SuppressDraw
Tick
UnloadContent
Update
Events
Activated
Deactivated
Disposed
Exiting
Components
Game
LaunchParameters
LaunchParameters
Class
Dictionary<string, string>
GameTime
Class
Properties
ElapsedGameTime
IsRunningSlowly
TotalGameTime
GameComponentCollection
Sealed Class
Collection<IGameComponent>
Events
ComponentAdded
ComponentRemoved
IGameComponent
Interface
Methods
Initialize
GameComponent
GameComponentCollectionEventArgs
Class
EventArgs
IGameComponent
IUpdateable
IDisposable
GameComponent
Class
Properties
Enabled
UpdateOrder
Methods
Initialize
OnEnabledChanged
OnUpdateOrderChanged
Update
Events
Disposed
EnabledChanged
UpdateOrderChanged
IUpdateable
Interface
Properties
Enabled
UpdateOrder
Methods
Update
Events
EnabledChanged
UpdateOrderChanged
IDrawable
DrawableGameComponent
Class
GameComponent
Properties
DrawOrder
GraphicsDevice
Visible
Methods
Draw
LoadContent
OnDrawOrderChanged
OnVisibleChanged
UnloadContent
Events
DrawOrderChanged
VisibleChanged
IDrawable
Interface
Properties
DrawOrder
Visible
Methods
Draw
Events
DrawOrderChanged
VisibleChanged

DODATEK A.2: Imenski prostor Microsoft.Xna.Framework - matematika

Math
Assembly: Microsoft.Xna.Framework
Namespace: Microsoft.Xna.Framework

IEquatable<Vector2>

Vector2
Struct

- Fields
  - X
  - Y
- Properties
  - One
  - UnitX
  - UnitY
  - Zero
- Methods
  - Add (+ 1 overload)
  - Barycentric (+ 1 overload)
  - CatmullRom (+ 1 overload)
  - Clamp (+ 1 overload)
  - Distance (+ 1 overload)
  - DistanceSquared (+ 1 overload)
  - Divide (+ 3 overloads)
  - Dot (+ 1 overload)
  - Hermite (+ 1 overload)
  - Length
  - LengthSquared
  - Lerp (+ 1 overload)
  - Max (+ 1 overload)
  - Min (+ 1 overload)
  - Multiply (+ 3 overloads)
  - Negate (+ 1 overload)
  - Normalize (+ 2 overloads)
  - operator - (+ 1 overload)
  - operator * (+ 2 overloads)
  - operator / (+ 1 overload)
  - operator +
  - Reflect (+ 1 overload)
  - SmoothStep (+ 1 overload)
  - Subtract (+ 1 overload)
  - Transform (+ 7 overloads)
  - TransformNormal (+ 3 overloads)

IEquatable<Vector3>

Vector3
Struct

- Fields
  - X
  - Y
  - Z
- Properties
  - Backward
  - Down
  - Forward
  - Left
  - One
  - Right
  - UnitX
  - UnitY
  - UnitZ
  - Up
  - Zero
- Methods
  - Add (+ 1 overload)
  - Barycentric (+ 1 overload)
  - CatmullRom (+ 1 overload)
  - Clamp (+ 1 overload)
  - Cross (+ 1 overload)
  - Distance (+ 1 overload)
  - DistanceSquared (+ 1 overload)
  - Divide (+ 3 overloads)
  - Dot (+ 1 overload)
  - Hermite (+ 1 overload)
  - Length
  - LengthSquared
  - Lerp (+ 1 overload)
  - Max (+ 1 overload)
  - Min (+ 1 overload)
  - Multiply (+ 3 overloads)
  - Negate (+ 1 overload)
  - Normalize (+ 2 overloads)
  - operator - (+ 1 overload)
  - operator * (+ 2 overloads)
  - operator / (+ 1 overload)
  - operator +
  - Reflect (+ 1 overload)
  - SmoothStep (+ 1 overload)
  - Subtract (+ 1 overload)
  - Transform (+ 7 overloads)
  - TransformNormal (+ 3 overloads)

IEquatable<Vector4>

Vector4
Struct

- Fields
  - W
  - X
  - Y
  - Z
- Properties
  - One
  - UnitW
  - UnitX
  - UnitY
  - UnitZ
  - Zero
- Methods
  - Add (+ 1 overload)
  - Barycentric (+ 1 overload)
  - CatmullRom (+ 1 overload)
  - Clamp (+ 1 overload)
  - Distance (+ 1 overload)
  - DistanceSquared (+ 1 overload)
  - Divide (+ 3 overloads)
  - Dot (+ 1 overload)
  - Hermite (+ 1 overload)
  - Length
  - LengthSquared
  - Lerp (+ 1 overload)
  - Max (+ 1 overload)
  - Min (+ 1 overload)
  - Multiply (+ 3 overloads)
  - Negate (+ 1 overload)
  - Normalize (+ 2 overloads)
  - operator - (+ 1 overload)
  - operator * (+ 2 overloads)
  - operator / (+ 1 overload)
  - operator +
  - SmoothStep (+ 1 overload)
  - Subtract (+ 1 overload)
  - Transform (+ 15 overloads)

IEquatable<Quaternion>

Quaternion
Struct

- Fields
  - W
  - X
  - Y
  - Z
- Properties
  - Identity
- Methods
  - Add (+ 1 overload)
  - Concatenate (+ 1 overload)
  - Conjugate (+ 2 overloads)
  - CreateFromAxisAngle (+ 1 overload)
  - CreateFromRotationMatrix (+ 1 overload)
  - CreateFromYawPitchRoll (+ 1 overload)
  - Divide (+ 1 overload)
  - Dot (+ 1 overload)
  - Inverse (+ 1 overload)
  - Length
  - LengthSquared
  - Lerp (+ 1 overload)
  - Multiply (+ 3 overloads)
  - Negate (+ 1 overload)
  - Normalize (+ 2 overloads)
  - operator - (+ 1 overload)
  - operator * (+ 1 overload)
  - operator /
  - operator +
  - Slerp (+ 1 overload)
  - Subtract (+ 1 overload)

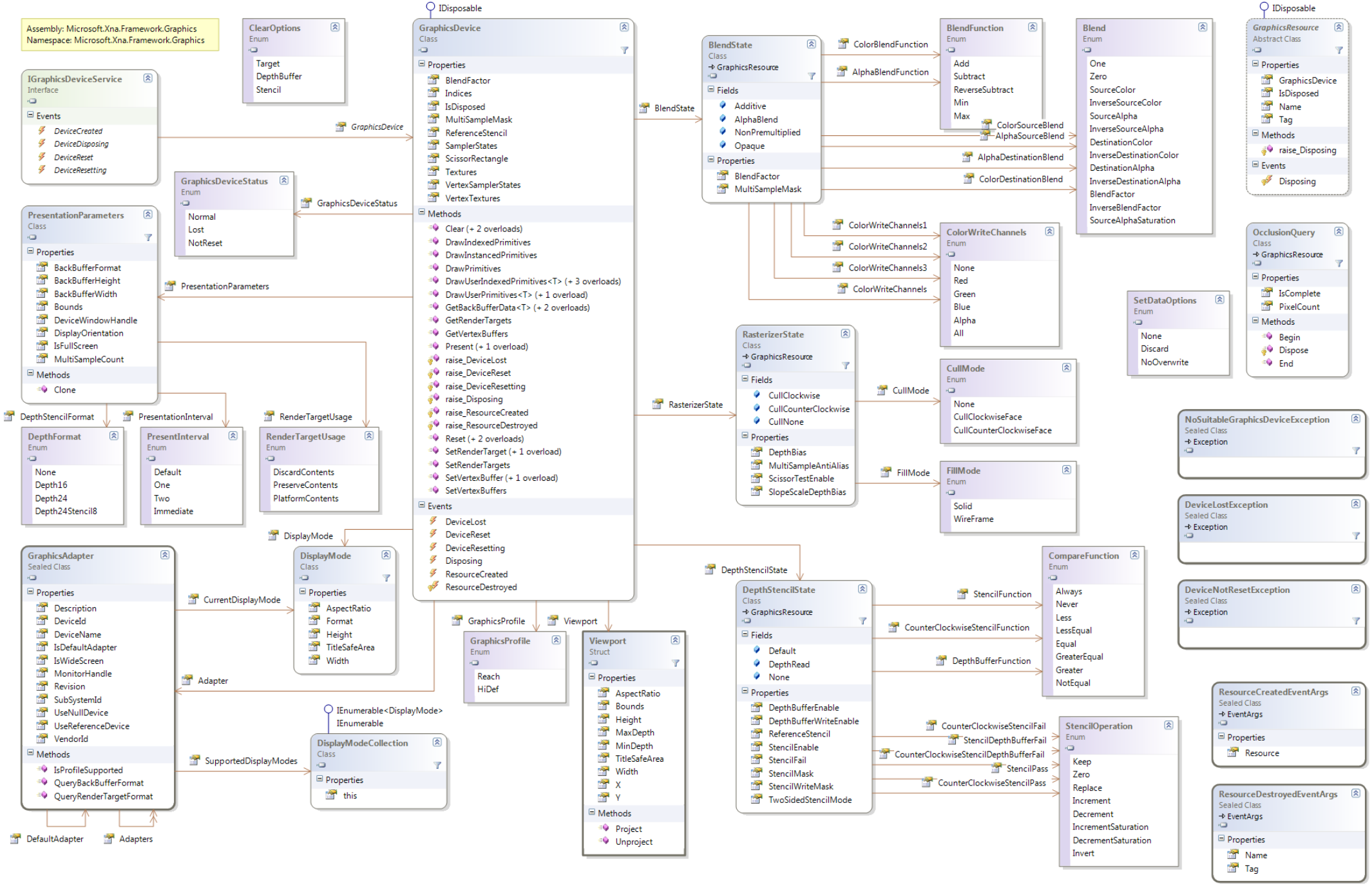
MathHelper
Static Class

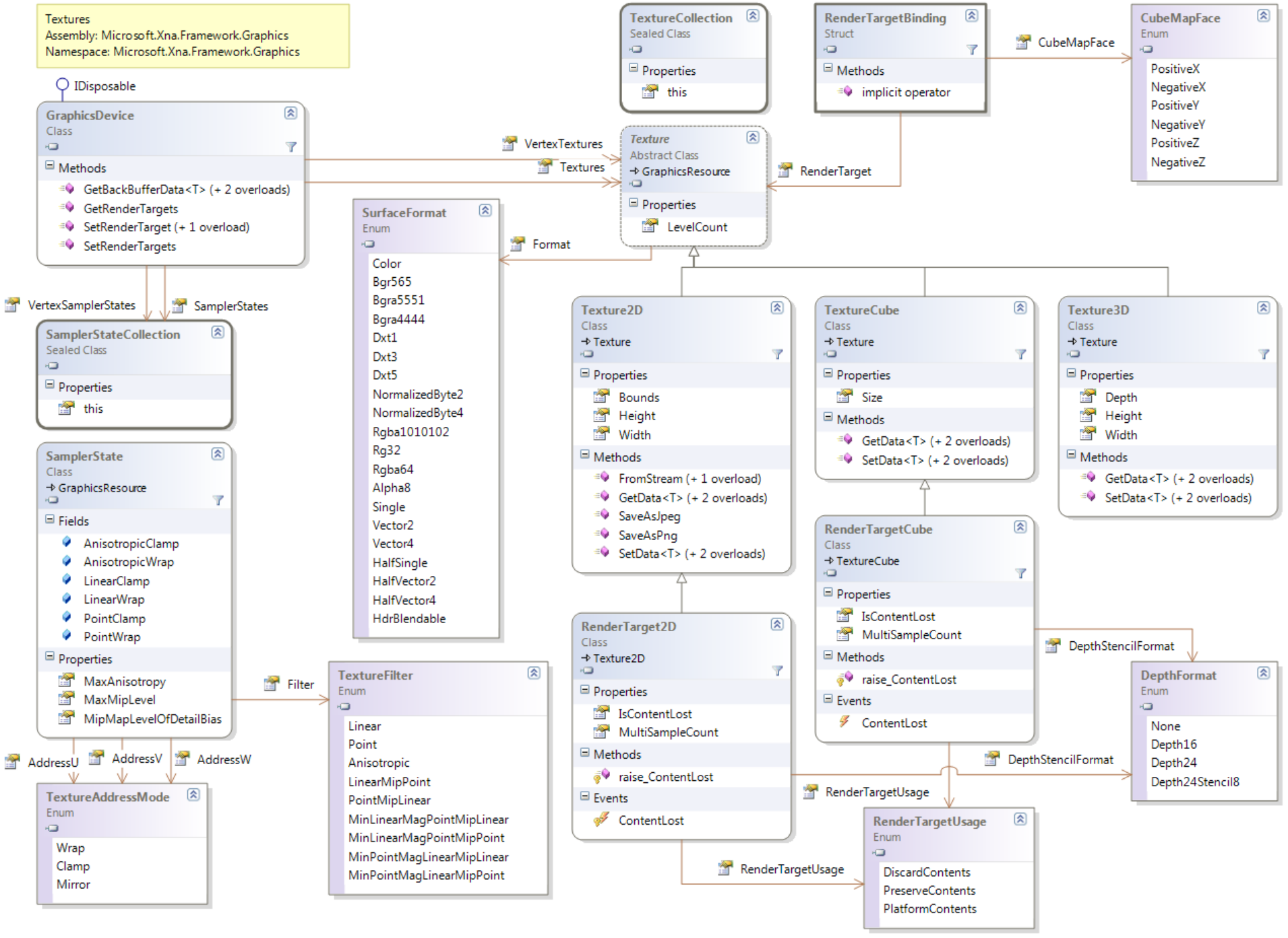
- Fields
  - E
  - Log10E
  - Log2E
  - Pi
  - PiOver2
  - PiOver4
  - TwoPi
- Methods
  - Barycentric
  - CatmullRom
  - Clamp
  - Distance
  - Hermite
  - Lerp
  - Max
  - Min
  - SmoothStep
  - ToDegrees
  - ToRadians
  - WrapAngle

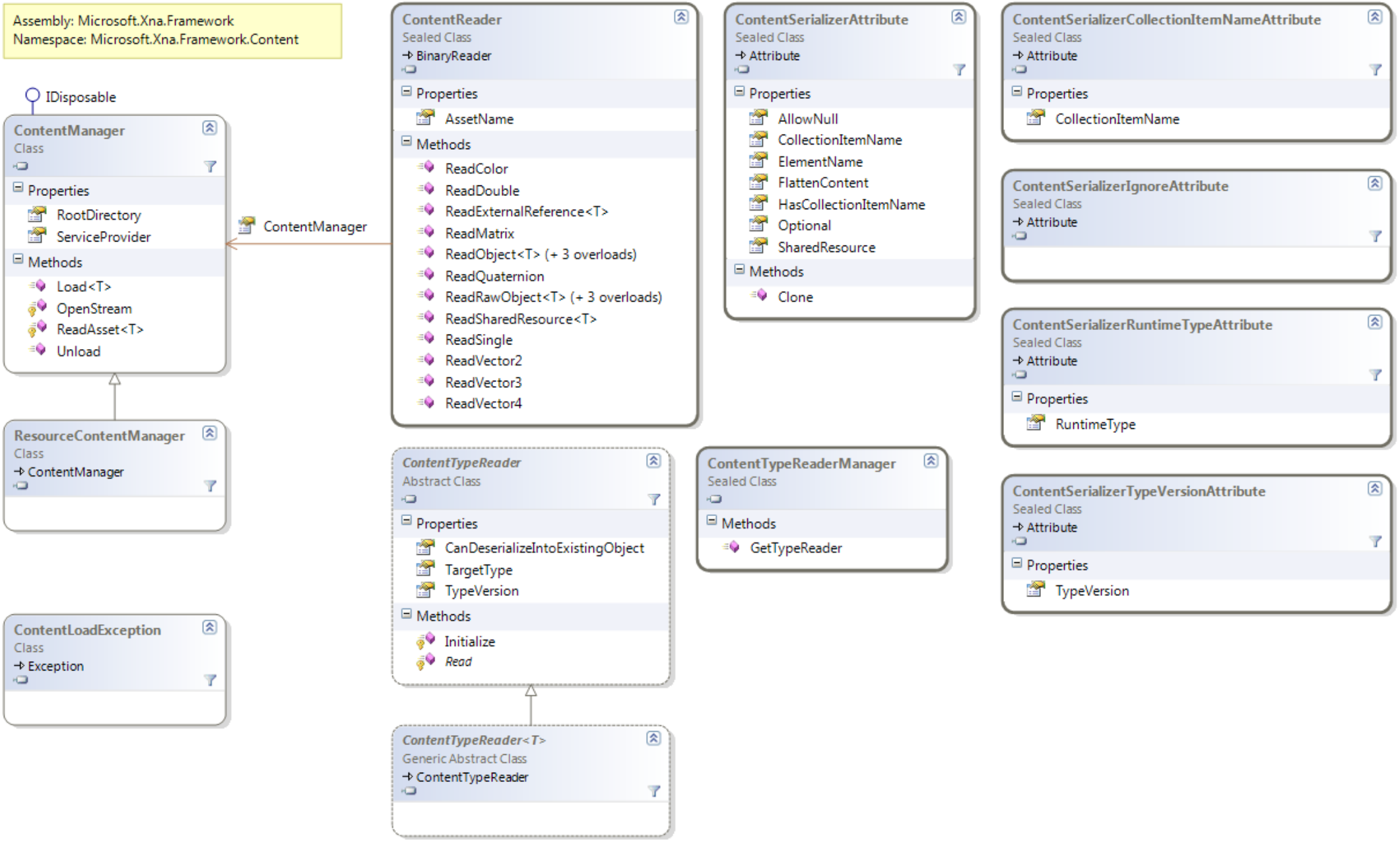
IEquatable<Matrix>

Matrix
Struct

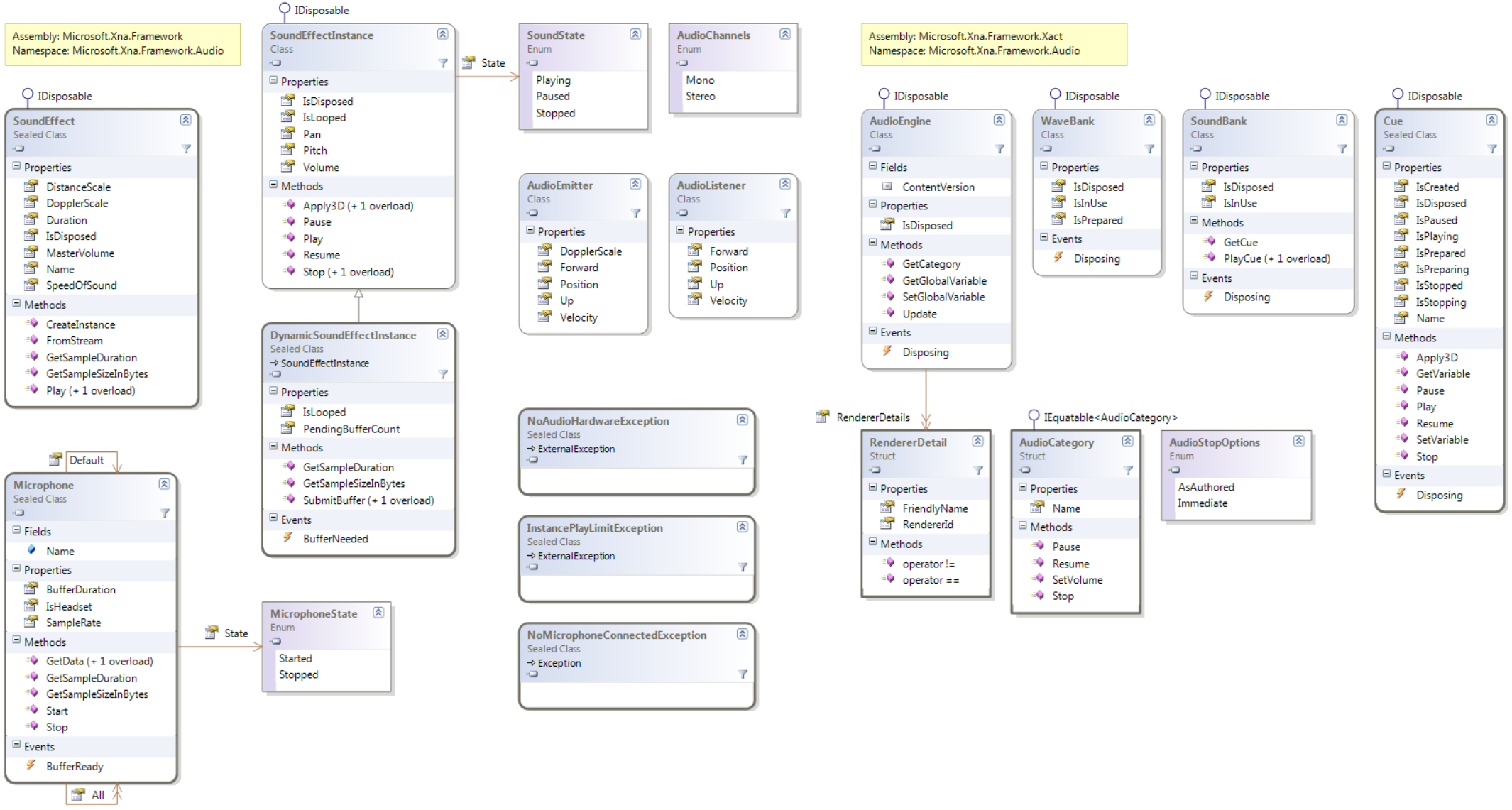
- Fields
  - M11
  - M12
  - M13
  - M14
  - M21
  - M22
  - M23
  - M24
  - M31
  - M32
  - M33
  - M34
  - M41
  - M42
  - M43
  - M44
- Properties
  - Backward
  - Down
  - Forward
  - Identity
  - Left
  - Right
  - Translation
  - Up
- Methods
  - Add (+ 1 overload)
  - CreateBillboard (+ 1 overload)
  - CreateConstrainedBillboard (+ 1 overload)
  - CreateFromAxisAngle (+ 1 overload)
  - CreateFromQuaternion (+ 1 overload)
  - CreateFromYawPitchRoll (+ 1 overload)
  - CreateLookAt (+ 1 overload)
  - CreateOrthographic (+ 1 overload)
  - CreateOrthographicOffCenter (+ 1 overload)
  - CreatePerspective (+ 1 overload)
  - CreatePerspectiveFieldOfView (+ 1 overload)
  - CreatePerspectiveOffCenter (+ 1 overload)
  - CreateReflection (+ 1 overload)
  - CreateRotationX (+ 1 overload)
  - CreateRotationY (+ 1 overload)
  - CreateRotationZ (+ 1 overload)
  - CreateScale (+ 5 overloads)
  - CreateShadow (+ 1 overload)
  - CreateTranslation (+ 3 overloads)
  - CreateWorld (+ 1 overload)
  - Decompose
  - Determinant
  - Divide (+ 3 overloads)
  - Invert (+ 1 overload)
  - Lerp (+ 1 overload)
  - Multiply (+ 3 overloads)
  - Negate (+ 1 overload)
  - operator - (+ 1 overload)
  - operator !=
  - operator * (+ 2 overloads)
  - operator / (+ 1 overload)
  - operator +
  - Subtract (+ 1 overload)
  - Transform (+ 1 overload)
  - Transpose (+ 1 overload)


Assembly: Microsoft.Xna.Framework.Graphics
Namespace: Microsoft.Xna.Framework.Graphics
IGraphicsDeviceService
Interface
Events
DeviceCreated
DeviceDisposing
DeviceReset
DeviceResetting
ClearOptions
Enum
Target
DepthBuffer
Stencil
GraphicsDevice
Class
Properties
BlendFactor
Indices
IsDisposed
MultiSampleMask
ReferenceStencil
SamplerStates
ScissorRectangle
Textures
VertexSamplerStates
VertexTextures
Methods
Clear (+ 2 overloads)
DrawIndexedPrimitives
DrawInstancedPrimitives
DrawPrimitives
DrawUserIndexedPrimitives<T> (+ 3 overloads)
DrawUserPrimitives<T> (+ 1 overload)
GetBackBufferData<T> (+ 2 overloads)
GetRenderTargets
GetVertexBuffers
Present (+ 1 overload)
raise_DeviceLost
raise_DeviceReset
raise_DeviceResetting
raise_Disposing
raise_ResourceCreated
raise_ResourceDestroyed
Reset (+ 2 overloads)
SetRenderTarget (+ 1 overload)
SetRenderTargets
SetVertexBuffer (+ 1 overload)
SetVertexBuffers
Events
DeviceLost
DeviceReset
DeviceResetting
Disposing
ResourceCreated
ResourceDestroyed
GraphicsDeviceStatus
Enum
Normal
Lost
NotReset
PresentationParameters
Class
Properties
BackBufferFormat
BackBufferHeight
BackBufferWidth
Bounds
DeviceWindowHandle
DisplayOrientation
IsFullScreen
MultiSampleCount
Methods
Clone
DepthFormat
Enum
None
Depth16
Depth24
Depth24Stencil8
PresentInterval
Enum
Default
One
Two
Immediate
RenderTargetUsage
Enum
DiscardContents
PreserveContents
PlatformContents
GraphicsAdapter
Sealed Class
Properties
Description
DeviceId
DeviceName
IsDefaultAdapter
IsWideScreen
MonitorHandle
Revision
SubSystemId
UseNullDevice
UseReferenceDevice
VendorId
Methods
IsProfileSupported
QueryBackBufferFormat
QueryRenderTargetFormat
DisplayMode
Class
Properties
AspectRatio
Format
Height
TitleSafeArea
Width
IEnumerable<DisplayMode>
IEnumerable
DisplayModeCollection
Class
Properties
this
GraphicsProfile
Enum
Reach
HiDef
Viewport
Struct
Properties
AspectRatio
Bounds
Height
MaxDepth
MinDepth
TitleSafeArea
Width
X
Y
Methods
Project
Unproject
BlendState
Class
GraphicsResource
Fields
Additive
AlphaBlend
NonPremultiplied
Opaque
Properties
BlendFactor
MultiSampleMask
BlendFunction
Enum
Add
Subtract
ReverseSubtract
Min
Max
Blend
Enum
One
Zero
SourceColor
InverseSourceColor
SourceAlpha
InverseSourceAlpha
DestinationColor
InverseDestinationColor
DestinationAlpha
InverseDestinationAlpha
BlendFactor
InverseBlendFactor
SourceAlphaSaturation
ColorWriteChannels
Enum
None
Red
Green
Blue
Alpha
All
RasterizerState
Class
GraphicsResource
Fields
CullClockwise
CullCounterClockwise
CullNone
Properties
DepthBias
MultiSampleAntiAlias
ScissorTestEnable
SlopeScaleDepthBias
CullMode
Enum
None
CullClockwiseFace
CullCounterClockwiseFace
FillMode
Enum
Solid
WireFrame
DepthStencilState
Class
GraphicsResource
Fields
Default
DepthRead
None
Properties
DepthBufferEnable
DepthBufferWriteEnable
ReferenceStencil
StencilEnable
StencilFail
StencilMask
StencilWriteMask
TwoSidedStencilMode
CompareFunction
Enum
Always
Never
Less
LessEqual
Equal
GreaterEqual
Greater
NotEqual
StencilOperation
Enum
Keep
Zero
Replace
Increment
Decrement
IncrementSaturation
DecrementSaturation
Invert
GraphicsResource
Abstract Class
Properties
GraphicsDevice
IsDisposed
Name
Tag
Methods
raise_Disposing
Events
Disposing
OcclusionQuery
Class
GraphicsResource
Properties
IsComplete
PixelCount
Methods
Begin
Dispose
End
SetDataOptions
Enum
None
Discard
NoOverwrite
NoSuitableGraphicsDeviceException
Sealed Class
Exception
DeviceLostException
Sealed Class
Exception
DeviceNotResetException
Sealed Class
Exception
ResourceCreatedEventArgs
Sealed Class
EventArgs
Properties
Resource
ResourceDestroyedEventArgs
Sealed Class
EventArgs
Properties
Name
Tag
GraphicsDevice
GraphicsDeviceStatus
PresentationParameters
DepthStencilFormat
PresentationInterval
RenderTargetUsage
DisplayMode
CurrentDisplayMode
Adapter
SupportedDisplayModes
DefaultAdapter
Adapters
GraphicsProfile
Viewport
BlendState
ColorBlendFunction
AlphaBlendFunction
ColorSourceBlend
AlphaSourceBlend
AlphaDestinationBlend
ColorDestinationBlend
ColorWriteChannels1
ColorWriteChannels2
ColorWriteChannels3
ColorWriteChannels
RasterizerState
CullMode
FillMode
DepthStencilState
StencilFunction
CounterClockwiseStencilFunction
DepthBufferFunction
CounterClockwiseStencilFail
StencilDepthBufferFail
CounterClockwiseStencilDepthBufferFail
StencilPass
CounterClockwiseStencilPass

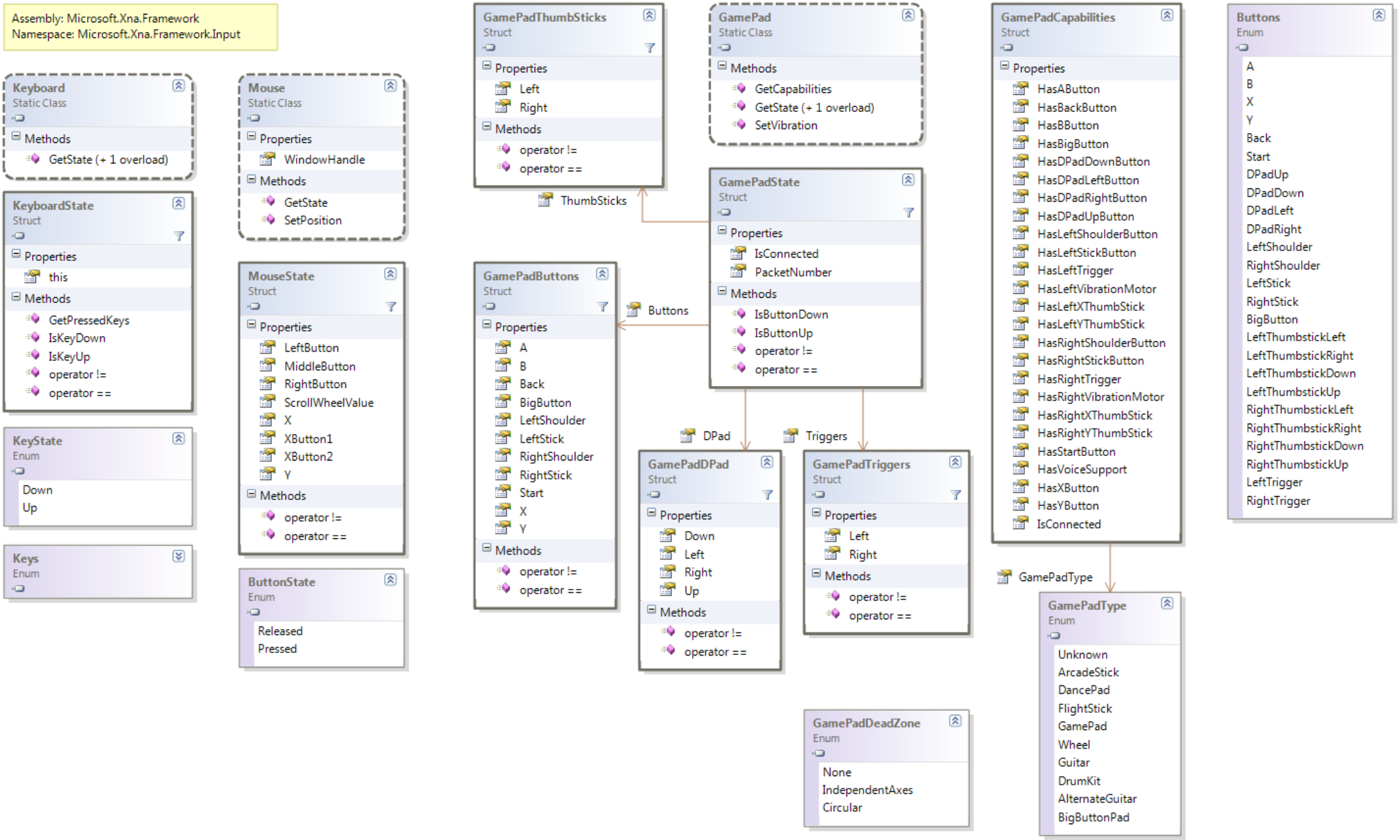

Textures
Assembly: Microsoft.Xna.Framework.Graphics
Namespace: Microsoft.Xna.Framework.Graphics
IDisposable
GraphicsDevice
Class
Methods
GetBackBufferData<T> (+ 2 overloads)
GetRenderTargets
SetRenderTarget (+ 1 overload)
SetRenderTargets
VertexTextures
Textures
Texture
Abstract Class
GraphicsResource
Properties
LevelCount
RenderTarget
TextureCollection
Sealed Class
Properties
this
RenderTargetBinding
Struct
Methods
implicit operator
CubeMapFace
CubeMapFace
Enum
PositiveX
NegativeX
PositiveY
NegativeY
PositiveZ
NegativeZ
SurfaceFormat
Enum
Color
Bgr565
Bgra5551
Bgra4444
Dxt1
Dxt3
Dxt5
NormalizedByte2
NormalizedByte4
Rgba1010102
Rg32
Rgba64
Alpha8
Single
Vector2
Vector4
HalfSingle
HalfVector2
HalfVector4
HdrBlendable
Format
VertexSamplerStates
SamplerStates
SamplerStateCollection
Sealed Class
Properties
this
SamplerState
Class
GraphicsResource
Fields
AnisotropicClamp
AnisotropicWrap
LinearClamp
LinearWrap
PointClamp
PointWrap
Properties
MaxAnisotropy
MaxMipLevel
MipMapLevelOfDetailBias
Filter
AddressU
AddressV
AddressW
TextureAddressMode
Enum
Wrap
Clamp
Mirror
TextureFilter
Enum
Linear
Point
Anisotropic
LinearMipPoint
PointMipLinear
MinLinearMagPointMipLinear
MinLinearMagPointMipPoint
MinPointMagLinearMipLinear
MinPointMagLinearMipPoint
Texture2D
Class
Texture
Properties
Bounds
Height
Width
Methods
FromStream (+ 1 overload)
GetData<T> (+ 2 overloads)
SaveAsJpeg
SaveAsPng
SetData<T> (+ 2 overloads)
TextureCube
Class
Texture
Properties
Size
Methods
GetData<T> (+ 2 overloads)
SetData<T> (+ 2 overloads)
Texture3D
Class
Texture
Properties
Depth
Height
Width
Methods
GetData<T> (+ 2 overloads)
SetData<T> (+ 2 overloads)
RenderTargetCube
Class
TextureCube
Properties
IsContentLost
MultiSampleCount
Methods
raise_ContentLost
Events
ContentLost
DepthStencilFormat
DepthFormat
Enum
None
Depth16
Depth24
Depth24Stencil8
RenderTarget2D
Class
Texture2D
Properties
IsContentLost
MultiSampleCount
Methods
raise_ContentLost
Events
ContentLost
RenderTargetUsage
RenderTargetUsage
Enum
DiscardContents
PreserveContents
PlatformContents


Assembly: Microsoft.Xna.Framework
Namespace: Microsoft.Xna.Framework.Content
IDisposable
ContentManager
Class
Properties
RootDirectory
ServiceProvider
Methods
Load<T>
OpenStream
ReadAsset<T>
Unload
ContentManager
ResourceContentManager
Class
ContentManager
ContentLoadException
Class
Exception
ContentReader
Sealed Class
BinaryReader
Properties
AssetName
Methods
ReadColor
ReadDouble
ReadExternalReference<T>
ReadMatrix
ReadObject<T> (+ 3 overloads)
ReadQuaternion
ReadRawObject<T> (+ 3 overloads)
ReadSharedResource<T>
ReadSingle
ReadVector2
ReadVector3
ReadVector4
ContentTypeReader
Abstract Class
Properties
CanDeserializeIntoExistingObject
TargetType
TypeVersion
Methods
Initialize
Read
ContentTypeReader<T>
Generic Abstract Class
ContentTypeReader
ContentSerializerAttribute
Sealed Class
Attribute
Properties
AllowNull
CollectionItemName
ElementName
FlattenContent
HasCollectionItemName
Optional
SharedResource
Methods
Clone
ContentTypeReaderManager
Sealed Class
Methods
GetTypeReader
ContentSerializerCollectionItemNameAttribute
Sealed Class
Attribute
Properties
CollectionItemName
ContentSerializerIgnoreAttribute
Sealed Class
Attribute
ContentSerializerRuntimeTypeAttribute
Sealed Class
Attribute
Properties
RuntimeType
ContentSerializerTypeVersionAttribute
Sealed Class
Attribute
Properties
TypeVersion

DODATEK A.6: Imenski prostor Microsoft.Xna.Audio



DODATEK A.7: Imenski prostor Microsoft.Xna.Input



Slike

Literatura

[1] Microsoft XNA, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Microsoft_XNA (datum zadnjega obiska: 15. 01. 2013)

[2] B. Nitschke, *Professional XNA Game Programming: For Xbox 360 and Windows*, Wiley Publishing, ZDA, 2007, pogl. 1.

[3] T. Miller,D.Johnson, *XNA Game Studio 4.0 Programming: Developing for Windows Phone 7 and Xbox 360*, Pearson Education, ZDA, 2011, pogl. Foreword, 3, 9.

[4] A. Ho, Albert Ho's XNA Redux, MSDN Blogs, dostopno na: http://blogs.msdn.com/b/al_msft/archive/2006/03/20/555065.aspx (datum zadnjega obiska: 21. 01. 2013)

[5] XNA Game Studio Team Blog, MSDN Blogs, dostopno na: http://blogs.msdn.com/b/xna/archive/2006/08/25/724607.aspx (datum zadnjega obiska: 21. 01. 2013)

[6] XNA Game Evolution Blog, dostopno na: http://evoxnagame.wordpress.com/2010/08/03/ready-start-go-%E2%80%93-xna-from-the-beginning-overview/ (datum zadnjega obiska: 11. 09. 2013)

[7] What's new in this release (XNA Game Studio 2.0), MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-us/library/bb417503(v=xnagamestudio.20).aspx (datum zadnjega obiska: 21. 01. 2013)

[8] Xbox Live, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Xbox_Live (datum zadnjega obiska: 21. 01. 2013)

[9] Games For Windows Live, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Games_for_Windows_-_Live (datum zadnjega obiska: 21. 01. 2013)

[10] What's new in this release (XNA Game Studio 3.0), MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-US/library/bb417503(v=xnagamestudio.30).aspx (datum zadnjega obiska: 23. 01. 2013)

[11] What's new in this release (XNA Game Studio 3.1), MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-us/library/bb417503(v=xnagamestudio.31).aspx (datum zadnjega obiska: 23. 01. 2013)

[12] What's New in XNA Game Studio 4.0, MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-us/library/hh221584.aspx (datum zadnjega obiska: 23. 01. 2013)

[13] What's New in XNA Game Studio 4.0 Refresh, MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-us/library/bb417503.aspx (datum zadnjega obiska: 23. 01. 2013)

[14] The XNA Framework Content Pipeline, XNA Game Studio Team Blog, MSDN Blogs, dostopno na: http://blogs.msdn.com/b/xna/archive/2006/08/29/730168.aspx (datum zadnjega obiska: 25. 01. 2013)

[15] What is content ?, MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-us/library/bb4756.aspx (datum zadnjega obiska: 25. 01. 2013)

[16] What is the Content Pipeline?, MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-us/library/bb447745.aspx (datum zadnjega obiska: 25. 01. 2013)

[17] Windows Phone 8 Usage Surges Into 2013, SFGate, dostopno na: http://www.sfgate.com/business/article/Windows-Phone-8-Usage-Surges-Into-2013-4326776.php (datum zadnjega obiska: 20. 03. 2013)

[18] Windows Phone Blog, Windows.com, dostopno na: http://blogs.windows.com/windows_phone/b/windowsphone/archive/2012/03/28/marketplace-arrives-in-13-new-countries.aspx (datum zadnjega obiska: 20. 03. 2013)

[19] Cross-Platform Conditional Compilation Symbols, MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-us/library/dd282469.aspx (datum zadnjega obiska: 28. 01. 2013)

[20] Mishkin Faustini, Building a Cross-Platform Game on PC, Xbox, and WP7 with XNA Framework Game Studio, DZone, dostopno na: http://mobile.dzone.com/articles/dont-publish-building-cross (datum zadnjega obiska: 28. 01. 2013)

[21] ClickOnce Deployment Overview , MSDN Library, Microsoft, dostopno na: http://msdn.microsoft.com/en-us/library/142dbbz4(v=vs.80).aspx (datum zadnjega obiska: 28. 01. 2013)

[22] ClickOnce, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/ClickOnce (datum zadnjega obiska: 28. 01. 2013)

[23] Xbox Live Indie Games Faq, MSDN, dostopno na: http://xbox.create.msdn.com/en-US/home/faq (datum zadnjega obiska: 30. 01. 2013)

[24] Happy new year, xbox live indie games!, XNA Game Studio Developer Education, MSDN Blogs, dostopno na: http://blogs.msdn.com/b/xna/archive/2012/01/04/happy-new-year-xbox-live-indie-games.aspx (datum zadnjega obiska: 20. 03. 2013)

[25] Managed Code, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Managed_code (datum zadnjega obiska: 30. 01. 2013)

[26] Vinayak Garg, Tech Blog, dostopno na: http://vinayakgarg.wordpress.com/2011/08/27/features-missing-in-visual-studio-2010-express/ (datum zadnjega obiska: 30. 01. 2013)

[27] It's official: XNA is dead, GamaSutra, dostopno na: http://gamasutra.com/view/news/185894/Its_official_XNA_is_dead.php (datum zadnjega obiska: 14. 02. 2013)

[28] Game Engine, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Game_engine (datum zadnjega obiska: 04. 02. 2013)

[29] Software Framework, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Software_framework (datum zadnjega obiska: 04. 02. 2013)

[30] Panda3D, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Panda3D (datum zadnjega obiska: 04. 02. 2013)

[31] Ogre3D, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/OGRE (datum zadnjega obiska: 04. 02. 2013)

[32] JMonkey Engine, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/JMonkey_Engine (datum zadnjega obiska: 04. 02. 2013)

[33] Crystal Space, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Crystal_Space (datum zadnjega obiska: 04. 02. 2013)

[34] SDL, dostopno na: http://www.libsdl.org/ (datum zadnjega obiska: 04. 02. 2013)

[35] Simple And Fast Multimedia Library, dostopno na: http://www.sfml-dev.org/features.php (datum zadnjega obiska: 04. 02. 2013)

[36] RPG Maker, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Rpgmaker (datum zadnjega obiska: 06. 02. 2013)

[37] Rampant Games, http://rampantgames.com (datum zadnjega obiska: 12. 11. 2013)

[38] Game Maker, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Game_Maker (datum zadnjega obiska: 06. 02. 2013)

[39] Digital World, http://adithyaharun.blogspot.com/ (datum zadnjega obiska: 12. 11. 2013)

[40] MonoXNA, Google Code, dostopno na: http://code.google.com/p/monoxna/ (datum zadnjega obiska: 06. 02. 2013)

[41] MonoGame, GitHub Wiki, dostopno na: https://github.com/mono/MonoGame/wiki (datum zadnjega obiska: 06. 02. 2013)

[42] ANX.Framework, Codeplex, dostopno na: http://anxframework.codeplex.com (datum zadnjega obiska: 22. 03. 2013)

[43] Matej Jan, XNI, dostopno na: http://xni.retronator.com/post/2850676376/introduction (datum zadnjega obiska: 07. 02. 2013)

[44] M. Jan, Ogrodje za razvoj iger za iOS, diplomska naloga, Fakulteta za računalništvo in informatiko Univerze v Ljubljani, Slovenija, 2011, dostopno na: http://eprints.fri.uni-lj.si/1545/ (datum zadnjega obiska: 10. 11. 2013)

[45] Moby Games, dostopno na: http://www.mobygames.com (datum zadnjega obiska: 21. 02. 2013)

[46] Terraria, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Terraria (datum zadnjega obiska: 11. 02. 2013)

[47] Magicka, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Magicka (datum zadnjega obiska: 11. 02. 2013)

[48] RPGamer, dostopno na: http://www.rpgamer.com (Datum zadnjega obiska: 21. 02. 2013)

[49] Bastion, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Bastion_(video_game) (datum zadnjega obiska: 11. 02. 2013)

[50] VideoGamer, dostopno na: http://www.videogamer.com (datum zadnjega obiska: 21. 02. 2013)

[51] B.Foster, AI War: Fleet Command Review, SavyGamer, dostopno na: http://savygamer.co.uk/2010/11/29/ai-war-fleet-command-pc-review/ (datum zadnjega obiska: 11. 02. 2013)

[52] Game Debate, dostopno na: http://www.game-debate.com (datum zadnjega obiska: 21. 02. 2013)

[53] Model–view–controller, Wikipedia, dostopno na: http://en.wikipedia.org/wiki/Modelviewcontroller (datum zadnjega obiska: 10. 09. 2013)

[54] O.Chiang, The Master of Online Mayhem, Forbes, dostopno na: http://www.forbes.com/forbes/2011/0228/technology-gabe-newell-videogames-valve-online-mayhem.html (datum zadnjega obiska: 29. 03. 2013)

[55] DigitalRune Blog, dostopno na: http://www.digitalrune.com/Support/Blog/tabid/719/EntryId/49/XNA-4-0-Class-Diagrams.aspx (datum zadnjega obiska: 15. 10. 2013)