**МОНГОЛ УЛСЫН ШИНЖЛЭХ УХААН,
ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
КОМПЬЮТЕРИЙН ТЕХНИК, МЕНЕЖМЕНТИЙН СУРГУУЛЬ**

Цэвэлийн Мандах

3 ХЭМЖЭЭСТ ТОГЛООМЫН ХӨДӨЛГҮҮР

Мэргэжлийн индекс: D480101
Мэргэжлийн нэр: Програм хангамж

**Компьютерийн ухааны бакалаврын
зэрэг горилох бүтээл**

УЛААНБААТАР 2005

**МОНГОЛ УЛСЫН ШИНЖЛЭХ УХААН,
ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
КОМПЬЮТЕРИЙН ТЕХНИК, МЕНЕЖМЕНТИЙН СУРГУУЛЬ**

Цэвэлийн Мандах

3 ХЭМЖЭЭСТ ТОГЛООМЫН ХӨДӨЛГҮҮР

Мэргэжлийн индекс: D480101
Мэргэжлийн нэр: Програм хангамж

**Компьютерийн ухааны бакалаврын
зэрэг горилох бүтээл**

Эрдэм шинжилгээний удирдагч:
Магистр, багш М.Алтанзүрх

Улаанбаатар 2005 он

Гарчиг

ОРШИЛ

Компьютерын хүчин чадал, мэдээллийн технологи хөгжихийн хэрээр компьютер график хүчтэй хөгжиж түүнийг дагаад 3 хэмжээст бодит дүрслэлтэй компьютер тоглоом бидний амьдралын нэг хэсэг болжээ. 3 хэмжээст компьютер тоглоом нь зөвхөн зугаа цэнгэлээр зогсохгүй сургалт, танин мэдэхүйн утгаараа өргөн хэрэглэгдэх болжээ. Жишээ нь нисгэгчдийн сургалтын програм, жолооны сургалтын програм зэрэг 3 хэмжээст /цаашид 3d/ програмууд нь их хэмжээний хөрөнгө мөнгө, цаг хугацаа хэмнэх нь тодорхой. 10-аад жилийн өмнө бид 80486 процессор дээр DOOM гэдэг 3d тоглоомыг анх тоглож байсан бол одоогийн 3d тоглоомууд нь 3d компьютер график, 3d дуу чимээ, хиймэл оюун ухаан бүгд л бараг бодит амьдрал дээрхтэй тун ойр дөхөж очжээ. 3d амьд дүрслэлтэй тоглоомын хамгийн чухал зүйл нь 3D game engine гэж нэрлэгдэх кодууд байдаг. 3 хэмжээст тоглоомын хөдөлгүүр нь 3d тоглоомд хэрэглэгдэх 3d график өгөгдлийн бүтэц, 3 хэмжээст амьд дүрслэлийг дүрслэн харуулах кодууд, мөн түүнчлэн тоглоомын орчинг бүрдүүлэх камер, гэрэл, сүүдэр, удирдлага, хиймэл оюун ухаан, физикийн хуулиуд зэрэг тоглоомонд шаардлагатай хамгийн чухал зүйлүүдийг агуулсан кодуудаар зогсохгүй 3d тоглоомыг бүтээхэд шаардлагатай хэрэглээний програмуудын нийлмэл юм. Тоглоомыг бүтээхэд шаардлагатай хэрэглээний програмууд гэдэгт уг тоглоомонд ашиглагдах 2 хэмжээст зураг, 3 хэмжээст модель, 3d анимаци, чимээ, дууны эффектүүд, стерео хөгжим зэрэг мэдээллүүдээ үүсгэх, засварлах програмууд орно.

I БҮЛЭГ. ОНОЛЫН ХЭСЭГ

1.1 3 ХЭМЖЭЭСТ ТОГЛООМЫН ХӨДӨЛГҮҮРИЙН ТУХАЙ

3 хэмжээст тоглоомуудад 3 хэмжээст өгөгдлийн бүтэц, файлын формат, тоглоомын интерфейс, төрөл бүрийн API ашиглан дэлгэцэнд дүрслэх, программын давталт, гэрэл, сүүдэр, нар, гал, ус, шуурга, цас зэрэг эффектүүд, удирдлага, камер ба сүлжээний интерфейс бүхий л програмчлалаа тухайн ашиглаж байгаа 3 хэмжээст тоглоомын хөдөлгүүр дээр скрипт кодоор бичдэг. Өөрөөр хэлбэл 3 хэмжээст тоглоомын хөдөлгүүр нь тоглоомын програмчлалд скрипт програмчлалын хэл нь болно гэсэн үг. 3 хэмжээст компьютер тоглоом нь 3 хэмжээст тоглоомын хөдөлгүүрээрээ дамжин бүх API функцууд, OpenGL, DirectX функцууддаа ханддаг. Сайн 3 хэмжээст тоглоомын хөдөлгүүр ашигласан тоглоом дүрслэл сайтай, гацалдахгүй, зохион байгуулалттай болдог. Хүснэгт 1.1–т 2003 оны шилдэг 10-н 3 хэмжээст тоглоомын хөдөлгүүрийг үзүүлэв. Тоглоом гаргадаггүй, зөвхөн 3D Game Engine хийдэг корпорациуд ч олон байдаг. Төрөл бүрийн 3D Game Engine –ууд байдаг. Жишээ нь: Counter-Strike тоглоом нь Half-Life Engine дээр суурилагдан хийгдсэн. Walt Disney –н Pirates Of Caribbean тоглоомонд Storm гэдэг Engine –г ашигласан байдаг. Энэ Engine –н эх код нь компиляци хийгдээгүй, интерпретатор хэлээр түүнийг ажиллуулдаг, эх код дотор нь ямар ч өөрчлөлт оруулж болдогоор хийгдсэн байна.

Хүснэгт 1.1

	Нэр	**Зохиогч**	**API**	**Програмч-лалын хэл**	**Үйлдлийн систем**
1	Torque	Garage Games	OpenGL, DirectX	C/C++	Windows, Linux, MacOS
2	TV3D SDK 6	True Vision3D, LLCo	DirectX	C/C++, C#, Delphi, VB6, VB.NET	Windows

3	3D Game Studio	Conitec Datasystems	DirectX	C/C++, Delphi	Windows
4	Source	Valve	DirectX	C/C++	Windows
5	Cipher	RH Systems Ltd.	OpenGL	C/C++	Windows
6	Reality Factory	Wild Tangent	OpenGL, DirectX	C/C++	Windows
7	Unreal Engine 3	Epic Games Inc.	OpenGL, DirectX	C/C++	Windows, Linux, Xbox,Playstation, GameCube
8	Quest 3D	Act-3D B.V	DirectX	C/C++	Windows
9	Blitz 3D	Blitz Research Ltd	DirectX	Basic	Windows
10	AMP 3D	Slam Software and 4D Rulers	OpenGL	C/C++	Windows

Жишээ болгон Torque 3d Game Engine-н үзүүлэлтүүдийг доор үзүүлэв.

Хийсэн компани: GarageGames

Үйлдлийн систем: Windows, Linux, MacOS

Програмчлалын хэл: C/C++

График интерфейс: DirectX, OpenGL

Програмчлал: Объект хандалтат програмчлал

Скрипт: С хэлтэй төстэй өөрийн скрипттэй

Эдиторууд: World Editor, Terrain Editor, Terrain Generator, GUI Editor гэх 4 график эдитортой

Боломжууд: Физик, 2d, 3d дуу, 3d график анимаци, гэрэл сүүдэр, гал ус тэнгэр… г.м. олон боломжуудтай.

1.2 ХЭРЭГЛЭГЧИЙН ШААРДЛАГА

1.2.1 Хэрэглэгчийн тухай мэдээлэл ба үйл ажиллагааны онцлог

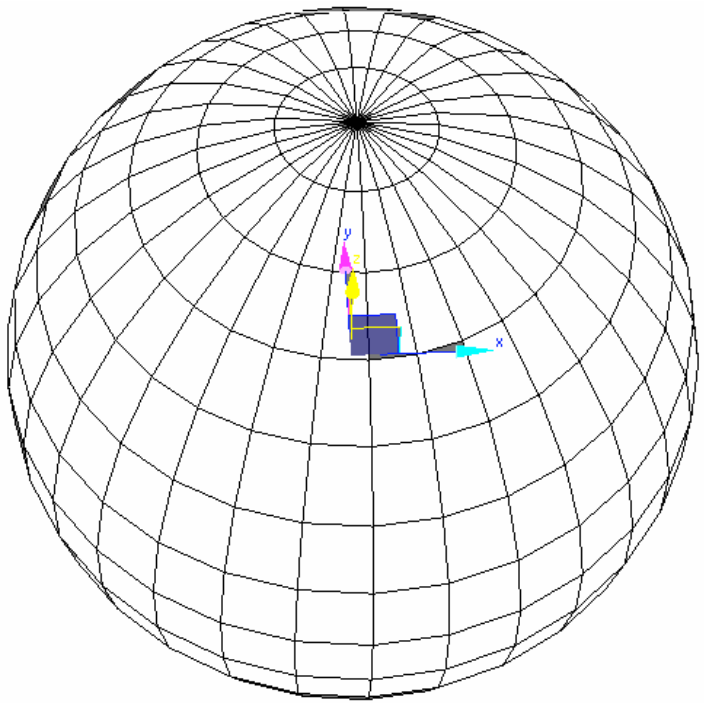
Бидний хэрэглэгчид бол бүхий л төрлийн 3 хэмжээст график програм зохиогчид юм.

1.2.2 Систем хөгжүүлэх үндэслэл

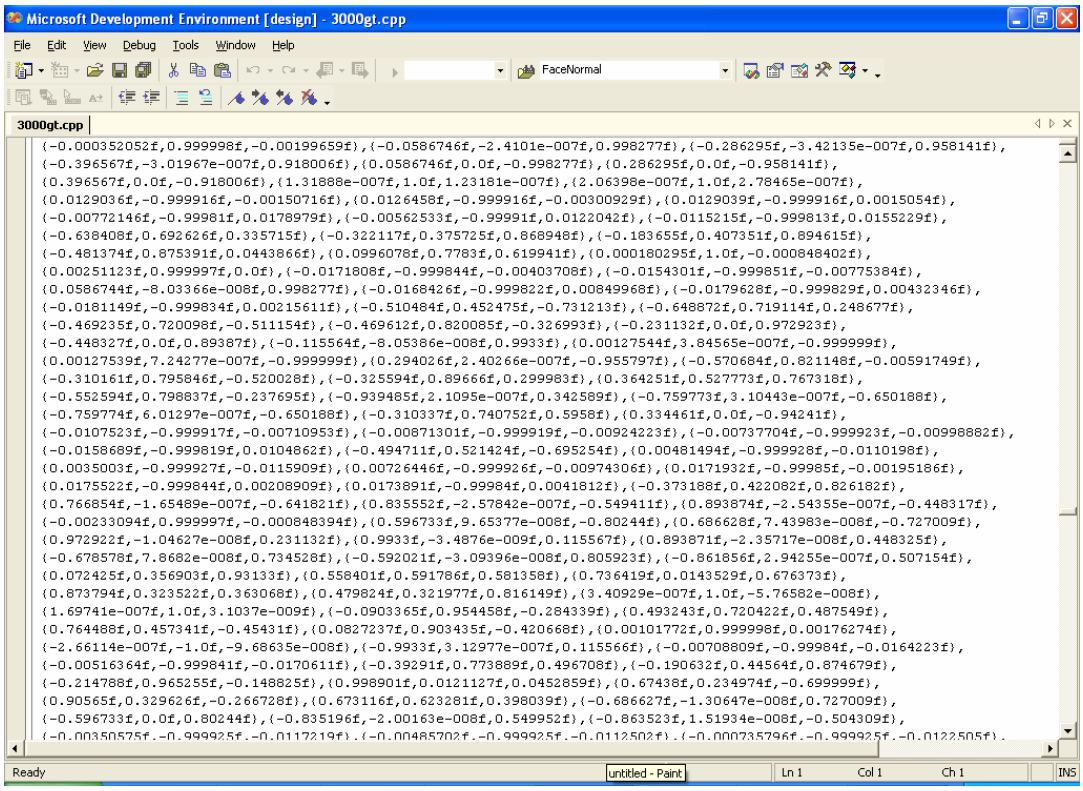
Програмистууд ямар нэгэн програмчлалын хэлээр 3 хэмжээст график, анимаци, дуу хөгжим, физик орчинтой програм бичихийн тулд уг програмдаа ашиглагдах дуу, дүрс зэрэг олон төрлийн мултимедиа өгөгдлүүдийг унших, бүтэцчилэн хадгалах, сэргээх зэрэг үйлдлүүдийг хийх 3 хэмжээст график хөдөлгүүр, 3 хэмжээст анимаци хөдөлгүүр, дуу дүрс, физик хөдөлгүүрүүд зайлшгүй хэрэгтэй болдог. Энэ бүгдийг багтаасан 3 хэмжээст тоглоомын хөдөлгүүрийг ашиглан програмаа бичвэл том хэмжээний мултимедиа өгөгдлийг ашиглаж буй үед ч хялбар ажил болох билээ.

Жишээ нь: Дунд зэргийн хэмжээтэй бөмбөлгийг маш цэвэрхэн яг бөмбөлөг мэт харагдуулахын тулд дор хаяж 32 сегменттэйгээр зурах хэрэг гарна. 32 сегменттэй бөмбөлөгт 482 ширхэг цэг, 960 ширхэг хавтгай оролцдог. Энэ бүх цэгүүдийн координатыг тооцоолон гаргах эсвэл эдгээрийг програмын эх код дотор хадгална гэвэл хэтэрхий бүдүүлэг болхи програм болох билээ. Геометрийн стандарт бус дүрсүүдийн хувьд бол бүүр бүтэшгүй хэрэг болно.

Дээрхи бүх мултимедиа өгөгдлүүдийг үүсгэх, хадгалах, унших, дүрслэх бүх үйлдлүүдийг хийдэг 3 хэмжээст тоглоомын хөдөлгүүр нь зайлшгүй чухал хэрэгцээтэй функцуудын сан юм.



Зураг 1.1 Бөмбөрцөгт оролцож буй цэг болон хавтгайнууд



Зураг 1.2 Цэгүүдийн координат болон хавтгайн тодорхойлогчдыг эх код дотроо агуулсан програм

1.2.3 Хэрэглэсэн нэр томъёоны тодорхойлолт

модель – 3 хэмжээст загвар дүрс

текстур – загвар дүрсийн гадуур бүрэх зураг

анимаци – загвар дүрсийн хөдөлгөөн

партикл – гал, ус зэргийн график эффектууд

эдитор програм – мултимедиа өгөгдлүүдийг үүсгэгч, засварлах бие даасан програм

меш бүтэц – 3 хэмжээст загвар дүрсийг илэрхийлэх нэгэн төрлийн мод бүтэц

фрейм – дэлгэцэнд зурагдаж буй нэг удаагийн зураг (кадр)

1.2.4 Хэрэглэгчийн функциональ шаардлага

Орчин үеийн 3 хэмжээст компьютер тоглоом, 3 хэмжээст тоглоомын хөдөлгүүрийн хувьд дараах зүйлсүүдийг шийдвэрлэсэн байх шаардлагатай.

- 3 хэмжээст моделийг файлаас унших, дэлгэцэнд дүрслэх
- 3 хэмжээст хөдөлгөөнт моделийг файлаас унших, дэлгэцэнд дүрслэх, дэлгэцэнд хөдөлгөөнийг дүрслэх
- 3 хэмжээст хөдөлгөөнт моделийг үүсгэх
- 2 хэмжээст график зургийг унших, дэлгэцэнд зурах, түүгээр 3 хэмжээст биеийг бүрэх
- Дууны файлтай ажиллах
- Видео файлтай ажиллах
- Физикийн орчинг бүрдүүлэх (Гравитацын хүч, биеийг мэдрэх, тулах физикийн хуулиудиар хангах)
- Камерыг удирдах
- Оролтын модулиар хангах

1.3 АРХИТЕКТУРЫН СОНГОЛТ

Уг 3 хэмжээст тоглоомын хөдөлгүүр нь дээрх хэрэгцээг хангаж өгсөн классуудын нэгдэл API (Class Library .DLL) болох юм.

1.3.1 Програмчлалын орчин, үйлдлийн систем

Нийтэд хүртээмжтэй байхын тулд олон хэрэглэгчтэй Windows үйлдлийн системыг сонгон авлаа. Энгийн хэрэглэгчдийн 90 орчим хувь нь Windows үйлдлийн систем ашигладаг.

Windows үйлдлийн системийн орчинд объект хандалтат програмчлалын хүчирхэг хэл C# -г сонгон авлаа. Visual C++/C# хэл нь Delphi, Java... гэх мэт бусад хэлүүдийг бодвол график application -д хамгийн тохиромжтой хэл юм. Нэгэнт объект хандалтат програмчлал ашиглах гэж байгаа тул C# -г сонгон авсан.

C#-н Reference Class Library .DLL файл болно.

1.3.2 3 хэмжээст график орчин

Компьютер тоглоом болон график программд 3 хэмжээст график орчинг хэд хэдэн аргаар бүрдүүлдэг.

- Software буюу программын аргаар
- WinG
- DirectX
- OpenGL

Ихэнх тоглоомууд эдгээр аргуудад бүгдэд нь тохируулан хийсэн, хэрэглэгч өөрөө ямар аргаар дүрслэхийг сонгож болдогоор хийгдсэн байдаг.

Программын аргаар орчинг бүрдүүлэхэд алслалт, гэрэл, материал, хувиргалт гэх мэт маш олон 3 хэмжээст орчны дүрмүүдийг өөрөө программчлан бичих хэрэгтэй болно. Хичнээн сайн алгоритм бичсэн гэсэн төв процессороор гүйцэтгүүлж байгаа учраас маш удаан болдог. Дэлхийн томоохон тоглоомын корпорациудын хийсэн тоглоомуудад ч гэсэн энэ аргыг ашиглан дүрслэхэд дүрс давхцах, нэвт харагдах, өнгө холилдох зэрэг алдаанууд байнга гардаг. Дэлгэцийн нягтшил, өнгөний гүнийг маш багаар тохируулсан ч нэг секундэд харагдах фреймийн тоо маш бага, эвтэйхэн харагдаж чаддаггүй. Иймээс төрөл бүрийн эффект, партикл, гал, ус, салхи, нэвт харагдалт зэргүүдийг хийх боломжгүй.

WinG хэмээх нэмэлт сангуудыг суулган 3 хэмжээст дүрсүүдийг үүсгэж болох боловч энэ нь бас л программын аргаар үүсгэж буй тул 3 хэмжээст тоглоомын орчныг бүрдүүлэхэд хангалтгүй байдаг. Ихэвчлэн 2 хэмжээст график дүрслэлтэй ба 3 хэмжээст графикыг хааяа л багахан хэрэглэдэг программуудад л WinG -г ашигласан харагддаг.

Microsoft DirectX нь тусгайлан тоглоомонд зориулж хийгдсэн маш хүчирхэг доод түвшний API ба үүнд Direct3D гэдэг 3 хэмжээст орчныг бүрдүүлэхэд зориулсан сан байдаг. Direct3D нь 3 хэмжээст орчинг 2 хэмжээст дэлгэцэнд дүрслэхэд зориулагдсан маш олон хувиргалт зөөлтийг хийх алгоритм, 3 хэмжээст орчны дүрмүүд болон бүтцүүд, гэрэл, сүүдэрийн эффект, партикл, гал, ус, утаа зэрэг бүх л зүйлсүүдийг агуулсан байдаг. График дүрслэл сайтай, дэлгэцийн кардыг нэг их голдоггүй.

OpenGL нь сайн дэлгэцийн кард дээр нэг секундэд гарах фреймын тоо нь маш өндөр. Иймээс маш сайн дүрслэлтэй, цэвэрхэн хөдөлгөөнтэй харагддаг. DirectX-н нэгэн адил 3 хэмжээст орчныг бүрдүүлэх хувиргалтууд, зөөлтүүдийг хийх матриц, алгоритм, 3 хэмжээст орчны дүрмүүд, гэрэл, сүүдэр, төрөл бүрийн эффектүүдийг багтаасан байдаг. Сул тал нь заавал OpenGL –г дэмждэг дэлгэцийн кард шаарддаг. Гэвч орчин үед дэлгэцийн кард нь компьютерийн нэгэн том чухал үзүүлэлт болсон, ихэнх компьютерүүдэд OpenGL –г дэмждэг дэлгэцийн кард байдаг учраас хамгийн хурдан ажиллах, цэвэрхэн дүрслэх OpenGL-г сонгон авлаа.

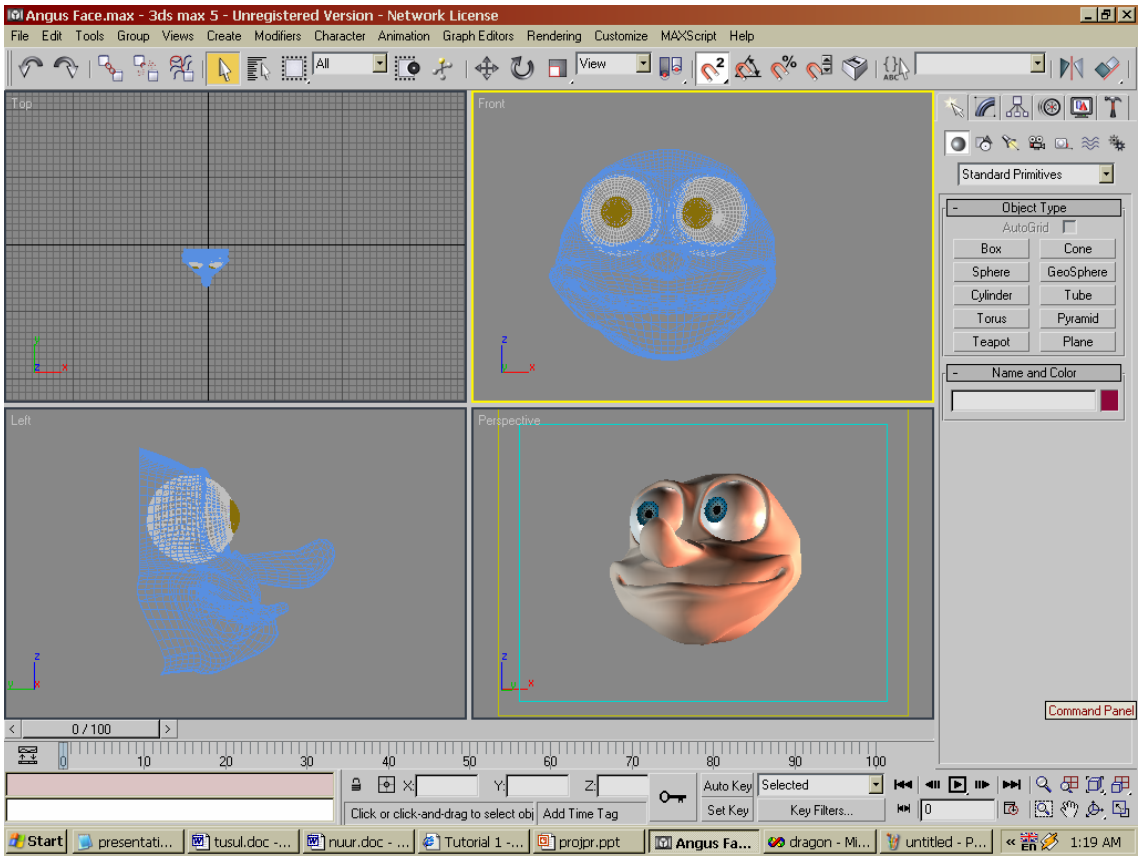
1.3.3 3 хэмжээст өгөгдлийн бүтэц

Тоглоомын эх код дотор моделиудын цэг бүрийн байрлал, гадаргуу бүрийн тодорхойлогчууд, текстурууд гэх мэтийн мэдээллүүдийг агуулахгүй байх хэрэгтэй. Үүнийг бүтэцчилэн файлд хадгалах хэрэгтэй.

Тэдгээрийн координатыг гараар гаргаж авах, хавтгайн байрлалыг тооцоолох гэдэг бол маш их хөдөлмөр хүч шаардсан ажил болно. Геометрийн стандарт дүрсүүдийг зурдаг алгоритм бичиж өгч болох ч ийм стандарт дүрсүүдээс бүрдсэн дүрслэл нь бодит амьдралтай нийцэхгүй, өнөө үеийн компьютер графикт шаардлага хангахааргүй болжээ. Харин график объектуудаа ямар нэгэн график эдитор программаар дэлгэцэнд харан хулганаараа зурж суух нь маш хялбар бас зугаатай ажил билээ. 3 хэмжээст дүрсийн эдитор программууд хэдийнээ өндөр түвшинд хүрсэн, бид ийм эдиторийг өөрсдөө бичих шаардлагагүй юм. 3D Studio Max, Caligari True Space гэх мэтийн өндөр үнэтэй дээд түвшний хэрэглээний програм хангамжууд олон байдаг. Тиймээс шууд л тэдгээр эдитор программуудаар график объектууд болон орчноо бүрдүүлээд өөрийнхөө программчлалд авч ашиглах нь зүйтэй.

Ингэснээр программын кодонд график объектууд байхгүй болж программын код бага хэмжээтэй, зохион байгуулалттай, улам боловсронгуй болно.

Олон 3 хэмжээст график эдитор программуудын олон янзын формат байдаг. Үүнээс хамгийн хүчирхэг 3D Studio Max программын .3ds төрөлтэй файл нь зөвхөн график объектуудыг л хадгалдаг, ямар нэг график эдитор программынх нь тохируулга ч гэх юмуу тиймэрхүү илүү зүйлсүүдийг агуулаагүй байдаг учраас бидэнд хамгийн тохиромжтой формат юм. Тиймээс энэ форматыг сонгож авлаа.



Зураг 1.3 3D Studio Max програм дээр модель зурж байгаа нь…

3ds файл нь өөрийн хувилбартай байдаг. Одоогоор өргөн хэрэглэгдэж байгаа нь 3.0 хувилбар. Энэ хувилбарын форматыг судлаж үзье.

3ds файл маш олон төрлийн бүтэцийг агуулдаг боловч энэ файлын нийтэд ил байдаггүй, зөвхөн эдитор программд л ашиглагддаг, энэ файлыг ашиглах гэсэн хүмүүс өөрсдөө энэ файлын форматыг судлан бүтцийг гаргадаг учраас олонхи бүтэцүүд нь тодорхой тайлагдаагүй байдаг. Бидний хувьд объектууд, объектын цэгүүдийн координат, гадаргуугын тодорхойлогчид, материал (текстур), хувиргалтын матрицууд зэргийн тухай мэдээлэл л байхад хангалттай. Тиймээс эдгээр зүйлсүүдийг л судлаж үзнэ.

1.3.4 3 хэмжээст модель анимаци болон түүний өгөгдлийн бүтэц

3 хэмжээст тоглоомд маш чухал асуудал бол 3 хэмжээст моделийг хөдөлгөх, түүнийг бүтэцчилэх асуудал байдаг. Доорх 2 янзаар бүтэцчилдэг.

- Цэг бүрийн координатыг фрейм бүрээр хадгалах
- Хөдөлгөөний дүрмийг зохиож моделио түүний дагуу хөдөлгөх

Хөдөлгөөний дүрэм зохиох олон янзын арга байдаг. Жишээ нь хүний болон амьтны модельд араг ясыг зурж түүнийхээ яс бүрийн хувиргалтын матрицыг фрейм бүрээр хадгалах. Програм ясны хувиргалтын матрицыг уншиж моделийн уг ясанд харъяалагдах цэг бүрийн координатыг тооцоолон гаргадаг. Жишээ нь хүний моделийн хувьд хөлний шилбэний ясны хувьд модель дээрх уг шилбэний ясанд харъяалагдах цэгүүдийг уг ясны хөдөлгөөний буюу хувиргалтын матрицаар хувиргана гэсэн үг. Зарим цэгүүд олон ясанд харъяалагдаж болно. Мөн түүнчлэн цэгүүд ясанд коефициентэт хамааралтай ч байж болно. Энэ арга нь цэг бүрийг хадгалсанаас санах ойн хувьд олон дахин хэмнэлттэй боловч програмын тооцооллын хувьд маш их дутагдалтай. Програм фрейм зурах бүртээ моделийн цэг бүрт хамгийн багадаа 1 удаа матрицын үржвэр бодох хэрэгтэй болно гэсэн үг. Харин цэгийн координат бүрийг фрейм бүрээр хадгалах нь санах ойн хувьд хэмнэлтгүй боловч тооцооллын хувьд бараг бодолт шаарддаггүй учраас энэ аргыг сонгон авлаа. Цэгүүдийн координатуудыг фрейм бүрээр нь агуулах .bla өргөтгөлтэй файлын төрөл зохиож ашиглая. Моделийн хөдөлгөөнийг цэгийн координатаар нь фрейм бүрээр нь .3ds төрөлтэй файлаас авч .bla форматтай файл үүсгэх “BLA File Creator” нэртэй хэрэгсэл програм бичиж энэ 3 хэмжээст тоглоомын хөдөлгүүртээ хавсаргая.

1.3.5 2 хэмжээст график буюу зургийн файлтай ажиллах ба текстур

3 хэмжээст компьютер тоглоомд 2 хэмжээст график файл нь 3 хэмжээст модельд текстур өгөх буюу моделийг зургаар бүрэх болон 2 хэмжээст зурган мэдээлэлд ашиглагдана.

Тусгай стандарт бус төрөлт файл ашиглахгүй. Нэвт харагдалтгүй графикт .bmp, нэвт харагддаг графикт шахалт хийгдээгүй .tga стандарт зургийн файл ашиглана. Иймээс дурын 2 хэмжээст график эдитор програмуудаар засварлах боломжтой ба тусгай зургийн эдитор програм бичих шаардлагагүй. Жишээ нь Adobe Photoshop, PaintShop… гэх мэт програмаар засварлах боломжтой. Эдгээр програмууд .bmp, .tga файлыг хоёуланг нь засварлах чадвартай.

1.3.6 Видео өгөгдөл

Тоглоомын танилцуулгыг гаргах ч юмуу хийсэн компанийн лого г гаргах гэдэг ч юмуу зарим тохиолдолд видео өгөгдөлтэй ажиллах хэрэгтэй болдог. Янз бүрийн стандарт болон стандарт бус видео өгөгдөл ашигладаг. Манай тохиолдолд windows стандарт видео файлын төрлийг ашиглая. ActiveX Player ашиглан дурын windows стандарт видео файлд хандаж болно.

1.3.7 Дууны файлтай ажиллах

Windows стандарт .wav файлын төрлийг ашиглая. Төрөл бүрийн эдитор програмууд байдаг учир тусгай засварлах програм бичих шаардлагагүй.

1.3.8 3 хэмжээст дууны эффект

Microsoft DirectX API ашиглан 3 хэмжээст дууны эффект оруулна. Стандарт .wav файлыг стерио спикерээр 3 хэмжээст орчин байгаа мэт дуугаргах бэлэн 3 хэмжээст дууны объект DirectSound API-д байдаг.

1.3.9 Камерийн дүрслэл

Камерийн байрлал, чиглэл, өнцөг, толгойг агуулсан, хөдөлгөөний функцууд, хувиргалтын матрицыг тооцолох класс бичиж өгнө. Уг камерийн классын төлвүүдээр камерийг чөлөөтэй хөдөлгөж болж байх ёстой.

1.3.10 Оролт буюу удирдлага

Оролт буюу удирдлага гар болон хулганаар явагдана. Microsoft DirectX-н Direct Input хүчирхэг API-г ашиглан оролтын төхөөрөмжид хандана. Товч дарсан, дараагүй, синхрон ба асинхрон төлөвтэй байх шаардлагатай. Синхрон гэдэг нь товчийг дарах яг тухайн нэг агшин дах төлөвийг хэлнэ. Асинхрон нь товчийг дарахаа зогсоох тухайн агшин дах төлөв юм. Тоглоомонд функциональ болон функциональ биш товчууд адилхан эрх тэгш зэрэглэлтэйгээр оролцдог. Хулганы хөдөлгөөний 3 хэмжээс (3 дах хэмжээс нь mouse wheel гэж нэрлэгддэг дээд талын гүйдэг зээрэнцэг), 3 товчний төлөвийг уншдаг байх шаардлагатай.

1.3.11 Физик хөдөлгүүр

Физик хөдөлгүүр нь тоглоомыг физикийн хуулиудаар хангаж өгч байдаг. Жишээ нь торх өндрөөс унах, машин хурдсан хөдлөх, юм мөргөх, хүн хананд тулах, сум хананд зоогдох гэх мэт… Зарим 3 хэмжээст тоглоомын

хөдөлгүүрүүд өөртөө физик хөдөлгүүр агуулсан, зарим нь агуулаагүй байдаг. Зөвхөн физик хөдөлгүүрийг хөгжүүлдэг компаниуд ч олон байдаг. Олонхи 3 хэмжээст тоглоомын хөдөлгүүрүүд бусад компанийн физик хөдөлгүүрүүдийг авч ашигладаг.

Манай тохиолдолд биет хананд тулах, сум хананд хүрэлцэх зэрэг физикийн хуулиудыг бичиж өгье. Харин бусад биеийн ойлт өнхрөлт зэргийг ODE физик хөдөлгүүр ашиглан тооцоолъё.

1.3.12 Сүлжээний интерфейс

Microsoft DirectX-н DirectPlay API ашиглан Client – Server байдлаар 2-8 хүн холбогдон тоглож болохоор зохион байгуулна. Client – Server холбогдох, мэдээлэл дамжуулах гэх зэрэг хэрэгцээтэй функцуудыг бичиж өгье.

1.4 ОНОЛЫН СУДАЛГААНЫ ХЭСЭГ

1.4.1 OpenGL

Бид нэгэнт OpenGL -р программчлахаар шийдсэн учир OpenGL-г яг юу хийдгийг нь нарийн сайн мэдэж авах хэрэгтэй.

OpenGL нь ерөнхийдөө дараах зүйлийг хийдэг.

- Объектыг байрлуулах буюу хувиргах
- Дэлгэцэнд харуулах

1.4.1.1 Объектыг хувиргах

Дэлгэцэнд объектыг дүрслэхэд дараах хувиргалт хийх хэрэгтэй болдог.

- Загвар хувиргалт

Компьютер тоглоом болон графикт ер нь дандаа хөдөлгөөнтэй дүрсүүд байдаг. Түүнийг өөрийнх нь координатын тэнхлэгээр эргүүлэх, томруулах болон жижигрүүлэх, зөөх, огтлох зэрэг үйлдлүүдийг хийж хөдөлгөөнд оруулдаг. Энэ боломжуудыг хангасан функцууд байдаг.

- Үзэгдэх хувиргалт

Бидний харах дэлгэц бол чөлөөтэй хөдлөх камер байх ёстой. Тиймээс объектуудыг бидэнд харагдах ёстой яг тэр талаас нь харж байгаа юм шиг хувиргаж зөөдөг.

1.4.1.2 Дэлгэцэнд дүрслэх

- Ар талыг хасах

Хавтгайн бидэнд харагдахгүй ар талыг хасаж өгдөг. Ингэснээр илүү хурдан ажиллагаатай болно.

- Проекцлох

3 хэмжээст орчинг 2 хэмжээст дэлгэц дээр проекцлох. Өөрөөр хэлбэл Z тэнхлэгийг хасна гэсэн үг.

- Харагдах хувиргалт

Бидний дэлгэцийн нягтшилд тохируулан координат хувиргалт хийх. Өөрөөр хэлбэл 10.1723 гэсэн цэгийг 10 болгож хувиргана гэсэн үг. Камерын харах төрлөөс хамааран бас өөр хувиргалтуудыг хийнэ.

1.4.1.3 OpenGL буферүүд

OpenGL -д 2 төрлийн буфер байдаг. Эдгээр буферүүдийг дэлгэцэнд дүрслэхэд ашигладаг.

- Өнгөний буфер

Өнгөний буферт дэлгэцэнд харагдах зүйлсийг зурдаг. 2 ширхэг байдаг. Эхнийхийг нь дэлгэцэнд харуулж байхад дараагийнх дээр нь зурж байдаг.

- Гүний буфер буюу Z-Буфер

Гүний буферт өнгөний буферт хадгалагдсан харгалзах цэгийн z координатыг хадгалдаг. Аливаа цэгийг проекцлохдоо урьд нь зурагдсан дүрсийн Z буферт хадгалагдаж байгаа гүнийг уншин хэрэв түүнээс бага гүнд байгаа цэгийг проекцлож байгаа бол түүнийг өнгөний буферт бичих зарчмаар зурна. Өөрөөр хэлбэл цаана байгаа цэгийг наана байгаа цэг дарж харагдах ёстой.

1.4.1.4 Хувиргалтууд болон хувиргалтын матриц

3 хэмжээст орчинд ямар нэг объектыг x, y, z тэнхлэгийн дагуу зөөх, эргүүлэх, сунгах зэрэг хувиргалтын матрицаар хувиргадаг. Хувиргалтыг хийхдээ цэгүүдийн координатыг хувиргалтын матрицаар үржүүлдэг.

Ингэснээр объект зөөлт, координатын эхийн хувьд суналт, координатын тэнхлэгүүдийн хувьд эргүүлэлт хийгдэнэ. 3 хэмжээст дүрсийн хэлбэрийг өөрчлөхгүйгээр хөдөлгөөнийг хийхдээ хугацааны агшин бүрт хувиргалтын матриц ашиглаж болдог.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} * \begin{bmatrix} a_{11}, a_{12}, a_{13}, a_{14} \\ a_{21}, a_{22}, a_{23}, a_{24} \\ a_{31}, a_{32}, a_{33}, a_{34} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11}*x+a_{12}*y+a_{13}*z+a_{14}, & a_{21}*x+a_{22}*y+a_{23}*z+a_{24}, & a_{31}*x+a_{32}*y+a_{33}*z+a_{34} \end{bmatrix}$$

Хувиргалт хийгдсэн цэгийн координат:

$x` = a_{11}*x+a_{12}*y+a_{13}*z+a_{14}$

$y` = a_{21}*x+a_{22}*y+a_{23}*z+a_{24}$

$z` = a_{31}*x+a_{32}*y+a_{33}*z+a_{34}$ болно.

Хувиргалт хийгдэхгүй нэгж матриц

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ байна.

Зөөлт, эргүүлэлт, сунгалтын матрицыг нэгж матрицаар үржүүлж хувиргалтын матрицыг гаргаж авдаг.

➢ Зөөлтийн матриц

Объектын байрлалыг хүссэн тэнхлэгийн дагуу шилжүүлэх. Зөөлтийн матриц дараах хэлбэртэй байна.

$$\begin{bmatrix} 1 & 0 & 0 & x` \\ 0 & 1 & 0 & y` \\ 0 & 0 & 1 & z` \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

x` – x тэнхлэгийн дагуух шилжилт

y` – y тэнхлэгийн дагуух шилжилт

z` – z тэнхлэгийн дагуух шилжилт

- Эргүүлэлтийн матриц

Объектыг хүссэн тэнхлэгийнхээ дагуу хүссэн өнцгөөр эргүүлэх. Эргүүлэлтийн матриц дараах хэлбэртэй байна.

x тэнхлэгийн дагуу Ө өнцгөөр эргүүлэх

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta) & -\sin(\Theta) & 0 \\ 0 & \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

y тэнхлэгийн дагуу Ө өнцгөөр эргүүлэх

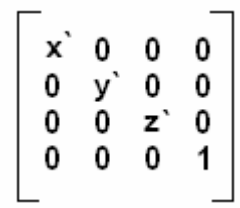
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta) & -\sin(\Theta) & 0 \\ 0 & \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

z тэнхлэгийн дагуу Ө өнцгөөр эргүүлэх

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta) & -\sin(\Theta) & 0 \\ 0 & \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Сунгалтын матриц

Объектын хэмжээг томруулах болон жижигрүүлэх. Сунгалтын матриц



x` – x тэнхлэгийн дагуух суналт

y` – y тэнхлэгийн дагуух суналт

z` – z тэнхлэгийн дагуух суналт

1.4.1.5 Материал оруулах

Ямар ч дүрсийг гурвалжин, дөрвөлжин болон олон өнцөгт хавтгай мэтээр дүрсэлдэг. Ингэхдээ шугамыг нь Брезенхемын алгоритмаар зурдаг. Энэ нь хамгийн хурдан алгоритм.

Гадаргууг дүрслэх хэд хэдэн арга байдаг.

- Хавтгайн цэгүүдийн тодорхойлогчыг харуулах

Хавтгайн цэгэн тодорхойлогчуудын координатыг дэлгэцэнд хэвлэх эсвэл зурж харуулах

- Шугамаар зурах

Хавтгайн тодорхойлогч цэгүүдийг холбох хооронд нь холбох

- Өнгөөр дүүргэх

Шугамаар зураад дотор нь будаг асгасан мэтээр дүүргэх

- Зургаар будах буюу текстур хийх

Зургийн файлаас эсвэл санах ойгоос зураг уншин хавгайд дүүргэх. Энэ нь маш сайхан харагддаг ба өнгөөр дүүргэснээс хурдан ажилладаг. Графикт ихэвчлэн үүнийг ашигладаг. Текстур өгөхдөө эхлээд санах ойд зургийн файлаа хийх, дараа нь уг материалаар цэг зурах болгондоо зургийн файлынхаа уг цэгт харгалзах координатыг өгөх маягаар зурна.

1.4.1.6 Дэлгэцэнд дүрслэх

Ерөнхийдөө дэлгэцэнд дүрслэхдээ дараах маягаар дүрслэнэ.

- Санах ой дахь бүтцээс бүх объектуудын хувьд бүх хавтгайн тодорхойлогч дахь цэгийн координатуудыг авч харгалзах материал өгөн объектын матрицаар хувиргалтыг хийн, хөдөлгөөний хувиргалтыг хийн зурна.
- Өнгөний буферийг солино (Дэлгэцэнд харуулах үйлдэл)

гэх маягаар давтна ...

1.4.1.7 OpenGL функцууд

glEnable(), glDisable() нь OpenGL –н зарим боломжуудыг хязгаарлах, хязгаарыг арилгах үйлдэл хийдэг функцууд юм.

```
void glEnable(GLenum cap);
void glDisable(GLenum cap);
```

Параметр *cap* нь OpenGL –н боломжуудын тогтмол. Олон тогтмолууд байдаг боловч бидэнд зайлшгүй хэрэгтэйг нь доор жагсаав.

GL_TEXTURE_2D – 2 хэмжээст зурган текстур. Текстур хэрэглэх бол заавал хязгаарлалтыг арилгах хэрэгтэй.

GL_DEPTH_TEST – Гүний буферыг харьцуулах, өөрчлөлт хийх. Хязгаарлалтыг арилгах хэрэгтэй.

```
void glViewport(GLint x,GLint y,GLsizei width,GLsizei height);
```

Энэ функцээр OpenGL –н зурах тэгш өнцөгтийг зааж өгдөг. x, y – зурах тэгш өнцөгтийн зүүн доод өнцөг, width, height –д тэгш өнцөгтийн өндөр, өргөнийг зааж өгдөг.

```
void gluPerspective(GLdouble fovy,GLdouble aspect,
                    GLdouble zNear,GLdouble zFar);
```

Дээрх функцээр алслалтыг тодорхойлно.

fovy –н у тэнхлэг дэх харах өнцөг (градусаар).

aspect – x тэнхлэгийн дагуу харах өнцгийн y тэнхлэгийнхтэй харьцуулсан харьцаа.

zNear – Ойрын огтлогч хавтгай хүртэлх зай. Өөрөөр хэлбэл энэ зайнаас нааш зүйлс харагдахгүй гэсэн үг.

zFar – Холын огтлогч хавтгай хүртэлх зай. Өөрөөр хэлбэл энэ зайнаас цааш зүйлс харагдахгүй.

```
void glMatrixMode(GLenum mode);
```

Энэ функцээр матрицыг зааж өгнө. mode – матрицын тогтмол.

GL_MODELVIEW -

GL_PROJECTION – проекцын матриц

GL_TEXTURE – текстур матриц

```
void glClearColor(GLclampf red,GLclampf green,
                  GLclampf blue,GLclampf alpha);
```

Дэлгэцийг цэвэрлэх өнгө. Өөрөөр хэлбэл дэлгэц цэвэрлэгдэхэд энэ өнгөөр дүүргэгдсэн байна.

```
void glClear(GLbitfield mask);
```

Буферийг цэвэрлэх буюу арилгах функц. mask – буферийн тогтмол.

GL_COLOR_BUFFER_BIT - өнгөний буфер

GL_DEPTH_BUFFER_BIT – Гүний буфер буюу z-буфер.

Логик үйлдэл ашиглан бичиж өгч болно.

```
BOOL SwapBuffers(HDC hdc);
```

Өнгөний буферүүдийг солих. hdc – handle to device context

OpenGl дээр ямар нэг дүрс зурахад тэр нь шууд дэлгэц дээр зурагддаггүй, хоёрдогч өнгөний буферт зурагддаг. Бүх объектуудаа зурсныхаа дараагаар өнгөний буферээ солих үйлдэл хийн хоёрдогч өнгөний буферээ дэлгэцэнд зурдаг. Үүний өмнө идэвхитэй байсан өнгөний буфер нь хоёрдогч буфер болно.

```
void glPushMatrix(void);
void glPopMatrix(void);
```

Эдгээр функцуудээр хувиргалтын матрицыг стекд хийх, стекээс гаргаж

авах үйлдлүүдийг хийнэ. Эдгээр функцуудыг ашигласнаар объект болгоноо өөр өөр матрицаар хувиргах, өөр өөрөөр хөдөлгөх зэрэг үйдлүүдийг хийх боломжтой болно.

```
void glMultMatrixf(const GLfloat *m);
```

Хувиргалтын матрицыг өгөгдсөн матрицаар үржүүлэх. m – матрицын заагч

```
void glColor3f(GLfloat red,GLfloat green,GLfloat blue);
```

Зурах дүрсийн өнгийг тодорхойлох функц. Өнгөний эрчимжилтийг өгнө.

```
void glColor4f(GLfloat red,GLfloat green,
                    GLfloat blue,GLfloat alpha);
```

Зурах дүрсийн өнгийг тодорхойлно. alpha – нэвт харагдалт

```
void glBindTexture(GLenum target,GLuint texture);
```

Зурах текстурыг сонгох.

target – зурах бай. 2 хэмжээст зурган текстурт GL_TEXTURE_2D байна.

texture – санах ойн хадгалагдсан текстур файлын дугаар буюу индекс.

```
void glBegin(GLenum mode);
void glEnd(void);
```

Эдгээр функцууд нь цэгүүдийг бүлэглэж өгдөг буюу ямар дүрс зурж буйг тодорхойлдог.

mode – зурах дүрсийн тогтмол

- GL_POINTS - Цэг
- GL_LINES - Шулуун
- GL_TRIANGLES - Гурвалжин
- GL_QUADS – Дөрвөн өнцөгт
- GL_POLYGON – Олон өнцөгт

```
void glVertex3f(GLfloat x,GLfloat y,GLfloat z);
```

Цэгийг тодорхойлох функц. glBegin(); glEnd(); функцуудын дунд бичигдэнэ. glBegin() функцын параметрт заагдсан дүрсийг зурна.

Жишээ нь: Гурвалжин зуръя гэвэл

```
glBegin(GL_TRIANGLES);
  glVertex3f(0,0,0);
  glVertex3f(0,1,1);
  glVertex3f(-1,0,1);
glEnd();
```

гэж дотор нь гурван цэг тодорхойлж өгнө.

void glTexCoord2f(GLfloat *s*,GLfloat *t*);
Тухайн зурах дүрсийг текстурээр бүрнэ. Параметрт нь тухайн тодорхойлох гэж буй цэгт харгалзах зургийн координатыг зааж өгнө.

1.4.2 .3ds файлын формат

3ds файл нь нэг буюу хэд хэдэн объектуудаар зохиогдсон 3 хэмжээст орчны хамгийн бага хэрэгтэй мэдээллүүдийн цувралыг агуулдаг. Ерөнхийдөө 3ds файл нь хэрчим буюу блокуудын жагсаалтыг агуулдаг. Энэ блокуудад юу байдаг вэ? Орчинг бүрдүүлэхэд зайлшгүй шаардлагатай бүх зүйлсүүд: бүх объектуудын: нэр, оройнуудын координат, гадаргуугын тодорхойлогч, материал, материалын координатууд, гадаргын өнгө, камер, гэрэл, хөдөлгөөний фреймүүд гэх мэт...

Блокууд нь нэг нэгнээсээ хамааралтай мод бүтэцтэй.

Блок нь 4 талбартай.

- Тодорхойлогч

2 байт 16-тын тоо ба блокыг тодорхойлдог. Энэ мэдээлэлээр ямар зүйлсийг агуулсан блок вэ? бидэнд хэрэгтэй хэрэггүйг мэдэж болно. Хэрэгтэй бол энэ блокыг уншина. Цааш нь хүүг нь уншина. Хэрэггүй бол алгасаж дараагийн блоц уруу үсэрнэ.

- Блокын урт

Энэ блок болон хүү блокуудын нийлбэр нийт урт 4 байт тоо.

- Мэдээлэл

  Хувьсах урттай блокын жинхэнэ мэдээллийг агуулах талбар.

- Хүү блокууд

  Энд хүү блокуудыг шууд агуулна.

Блок нь дараах хэлбэртэй байна гэсэн үг. /Хүснэгт 3.1/

Хүснэгт 3.1 Блок бүтэц

Эхлэл (байт)	Урт (байт)	Тайлбар
0	2	Тодорхойлогч
2	4	Блокын урт: мэдээлэл + хүү блокууд (6+n+m)
6	N	Мэдээлэл
6+n	M	Хүү блокууд

1.4.2.1 Блокуудын тайлбар жагсаалт

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4d4dh	M3Dmagic (*3ds files)	Байхгүй	M3D_Version, Mesh_Data, KeyFrameData	Агуулсан мэдээлэл байхгүй

3DS файлыг тодорхойлогч блок. Уг файлын бүтцийн эхэнд заавал байдаг.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
0002h	M3D_Version	M3Dmagic	Байхгүй	Short version;

3DS файлын хувилбарын дугаарыг агуулах блок. Хувилбарын дугаар нь short төрөлтэй байна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
3d3dh	Mesh_Data (*.3ds files)	M3Dmagic	Mesh_Version, Master_Scale, Object_Block, Mat_Entry	Байхгүй

Mesh-н талаархи мэдээлийг агуулах блок. Хүү блокуудаараа mesh-г тодорхойлно.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
3d3eh	Mesh_Version	Mesh_Data	Байхгүй	Байхгүй

Mesh-н хувилбарын дугаарыг агуулна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
0100h	Master_Scale	Mesh_Data	Байхгүй	Float scale;

Mesh-н ерөнхий суналтын коэффициентыг агуулна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4000h	Object_Block	Mesh_Data	Triangular_Mesh, Light, Camera	Cstr name;

Объектыг тодорхойлно. Объект бүрт буюу дүрс, гэрэл, камер бүрт Object_Block байна. Энэ блок нь уг объектын нэрийг болон уг объектын төрөл, хэлбэр, байрлал, материал зэргийг тодорхойлох блокуудыг агуулна. Объектын нэр нь ZSTR буюу тэг төгсгөлт тэмдэгт мөр байна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4100h	Triangular_Mesh	Object_Block	Vertices_List, Vertex_Flag_Array, Mesh_Matrix, Faces_Desc;	Байхгүй

Дүрсийг гурвалжингуудаар дүрслэх блок буюу объектын хэлбэр дүрсийн мэдээллийг агуулах блок.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4110h	Vertices_List	Triangular_Mesh	Байхгүй	Short n_points; Struct { Float x,y,z; } points[n_points];

Уг объектыг үүсгэж буй бүх цэгүүдийн координатуудыг агуулах блок. Цэгийн тоо, цэгийн x, y, z тэнхлэгүүдийн координатуудыг агуулах бүтэцүүдийн массивыг агуулна. Цэгүүдийн координат нь 3 3 хэмжээст орчин дахь x, y, z координат.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга

4120h	Faces_Desc	Triangular_Mesh	Smooth_Group	Short n_faces;
				Struct {
				Short vertex1, vertex2, vertex3;
				Short flags;
				}
				facearray[n_faces];

Гадаргуунуудын талаархи мэдээллийг агуулах блок. Гадаргуунуудын тоо болон гадаргуугын тодорхойлогч 3 цэгийн индексүүдийг агуулсан массивийг агуулна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4130h	Faces_Mat_List	Faces_Desc		Cstr material_name; Short nfaces; Short facenum[n_faces];

Гадаргуугын материалын тодорхойлогч. Материалын нэр болон уг материалтай гадаргуунуудын тоо болон индексүүдийг агуулна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4140h	Map_Coordinates List	Triangular_Mesh		Short n_verts; Struct { Float x,y; } vertices[nverts];

Гадаргуугын материалын UV координат. Гадаргууг бүрэх материалын цэг бүрт харгалзах цэгийн 2 хэмжээст зураг дээрх координатыг U, V координат гэдэг. Объектын цэг тус бүрт харгалзах материалын UV координатуудын массивыг агуулна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4160h	Mesh_Matrix (local coordinate system)	Triangular_Mesh		Float matrix[4][3];

Объектын хувиргалтын матриц. Уг хувиргалтын матрицд объектын шилжилт, эргэлт, суналт зэрэг мэдээллүүд агуулагдана.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4165h	Mesh_Color	Triangular_Mesh		Short color_index;

Объектын эдитор программууд дээр харагдах өнгө. Объектын өнгө гэж тусдаа зүйл байдаггүй. Зөвхөн материалаар өнгийг тодорхойлдог. Материалгүй бол өнгөгүй гэсэн үг. Энэ өнгө бол рендер хийхэд харагдах ёсгүй. Бид энэ блокыг унших шаардлагагүй.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
4170h	Mesh_Texture_Info	Triangular_Mesh		Short map_type; Float x_tiling, y_tiling; Float icon_x, icon_y, icon_z; Float matrix[4][3]; Float scaling, plan_icon_w, plan_icon_h, cyl_icon_h;

Объектын текстурын талаархи өргөтгөсөн мэдээлэл. Бид материалаар текстурыг тодорхойлох учир энэ блок бидэнд хэрэггүй.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
0010h	Color_F			Float red,green,blue;

Өнгийг RGB хөвөгч таслалтай тоогоор дүрслэх блок.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
0011h	Color_24			Char red,green,blue;

Өнгийг тэмдэгт төрөл буюу RGB 0-255 тоогоор дүрслэх блок.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
0030h	INT_Percentage			Short percentage;

Процент буюу хувин мэдээлэл. 0-100

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
0031h	FLOAT_Percentage			Float percentage;

Процент буюу хувин мэдээлэл. 0-1 хөвөгч таслалтай тоо.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
1100h	Background_Bitmap	Mesh_Data		Cstr filename;

Дэвсгэр зургийн блок. Дэвсгэр зургийн файлын нэрийг агуулна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
1200h	Background_Color	Mesh_Data	Color_F	

Дэвсгэрийн өнгө. Өнгөний блокыг агуулна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
Afffh	**Mat_Entry**	Mesh_Data	**Mat_Block,** Mat_Ambient, Mat_Diffuse, Mat_Specular, Mat_Shininess, Mat_Transparency, **Mat_TexMap**	

Материалын тодорхойлогч. Материалуудыг агуулдаг.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
a000h	**Mat_Block**	Mat_Entry		Cstr material_name;

Материал блок. Материал бүрт энэ блок байх ба материалын талаархи бүхий л мэдээллүүдийг агуулах блокуудыг агуулна.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
a010h	Mat_Ambient	**Mat_Block**	Color chunk	

Материалын амбиент өнгө буюу өөрөөс нь жигт ялгарч буй гэрлийн өнгө.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
a020h	Mat_Diffuse	**Mat_Block**	Color chunk	

Материалын ойлтын өнгө буюу материалын өнгө.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
a030h	Mat_Specular	**Mat_Block**	Color chunk	

Материалын сарнилтын өнгө.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
a040h	Mat_Shininess	**Mat_Block**	Percentage chunk	

Материалын тодролт.

ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
a050h	Mat_Transparency	**Mat_Block**	Percentage chunk	

Материалын нэвт харагдалт.

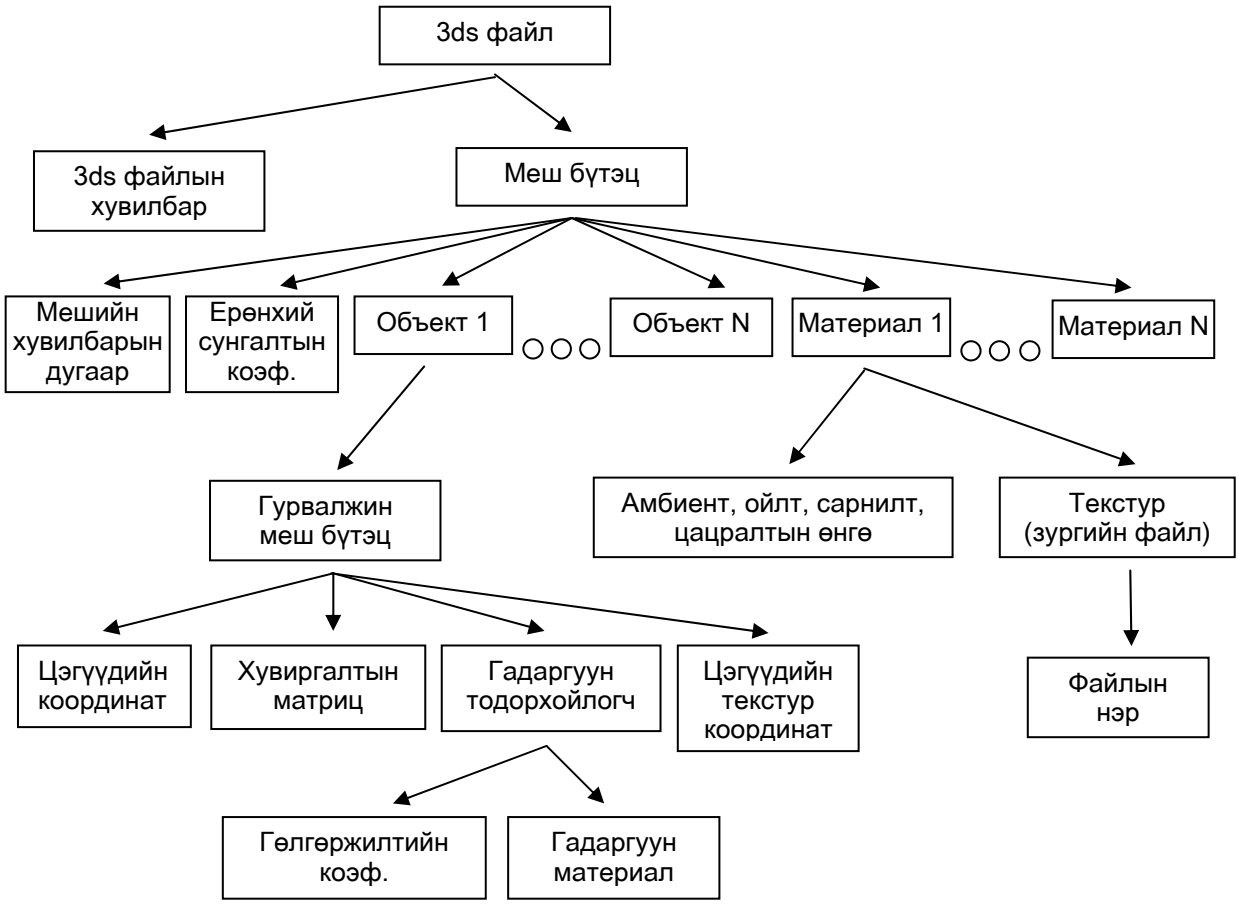
ID	Нэр	Эцэг блок	Хүү блокууд	Агуулга
a200h	**Mat_TexMap**	**Mat_Block**	Percentage chunk, Mat_MapName	

Материалын текстурын мэдээлэл. Материалын текстурын мэдээллийг агуулна.

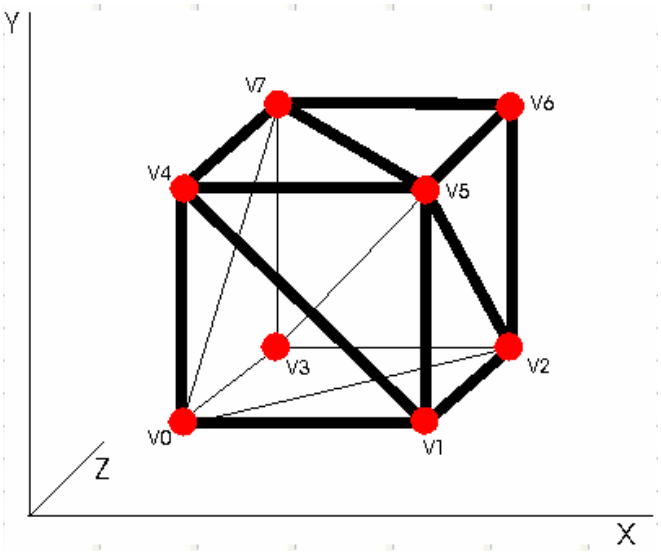
1.4.2.2 .3ds файлын мод бүтэц

Мод нь ерөнхийдөө доорх хэлбэртэй байх боловч агуулгаасаа хамааран өөр өөр байна. /Зураг 1.4/

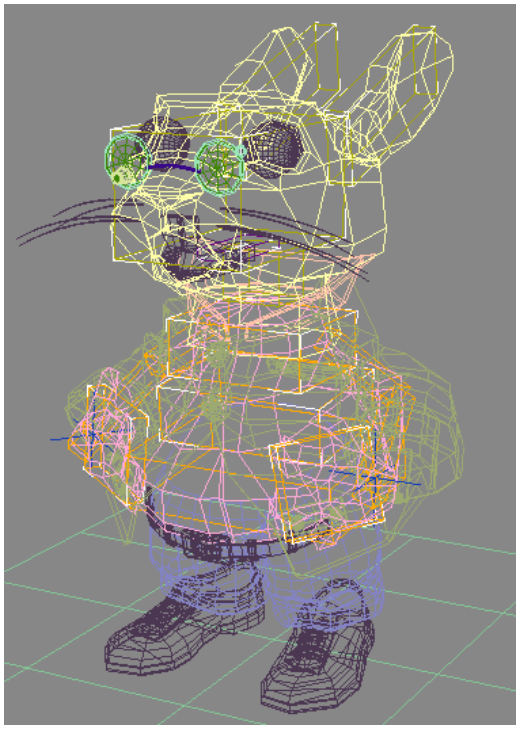
Зураг 1.4 .3ds файлын мод бүтэц



Бүх объектууд гурвалжин тор маягаар зурагддаг. Куб зуръя гэвэл бас л гурвалжин хавтгайнууд ашиглаж зурна. /Зураг 1.5, 1.6, 1.7/



Зураг 1.5 Энэ куб дүрсэнд 8 цэг, 12 гурвалжин хавтгай оролцож байна



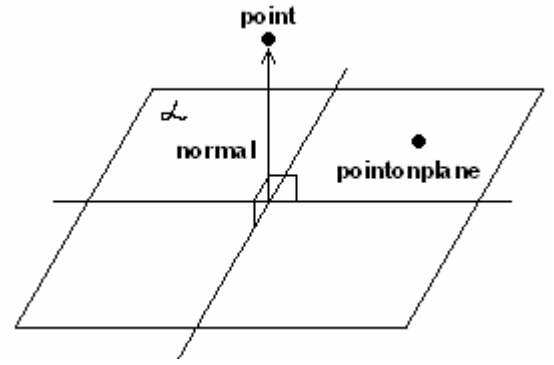
Зураг 1.6, 1.7 Бүх дүрс гурвалжин тор байдлаар зурагддаг

1.4.3 3D хэмжээст математик

Тоглоомын физикт хэрэглэгдэх 3d хэмжээст математик, геометрийн томъёонууд.

- Цэгээс хавтгай хүртэлх зай

point – цэгийн байрлал
normal, pointonplane – хавтгайн тодорхойлогчид
(normal – хавтгайн нормал вектор, pointonplane – хавтгай дээрх дурын цэг)



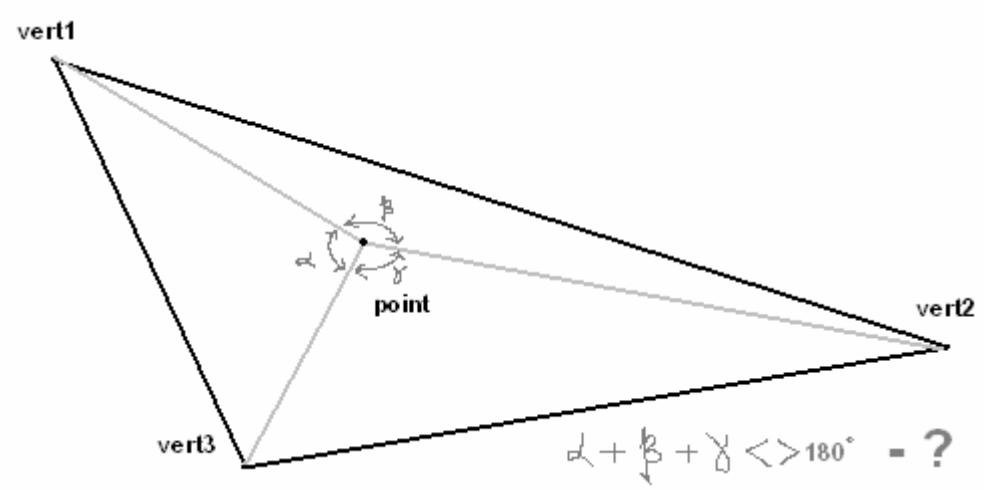
Зураг 1.8 Цэгээс хавтгай хүртэлх зай

normal * point – normal * pointonplane

- Тухайн цэг олон өнцөгт дээр оршиж байгаа эсэхийг шалгах

point – цэгийн координат
vert1, vert2, vert3 – олон өнцөгтийн оройн цэгүүдийн координатууд

$$\alpha + \beta + \gamma \langle\rangle 180^0$$



Зураг 1.9 Цэг олон өнцөгт дээр оршиж буй эсэх

- Хоёр векторын хоорондох өнцгийг олох

vec1, vec2 – векторын тодорхойлогчууд

```
acos(dot(vec1,vec2))
```

1.4.4 Физик хөдөлгүүр

1.4.4.1 Ойлголт

- Хатуу бие

Хатуу бие нь манай виртуал загварын зүгээс харахад хэд хэдэн шинж чанартай байдаг.

Зарим шинж чанарууд нь бүхий л хугацааны туршид өөрчлөгдөж байдаг.

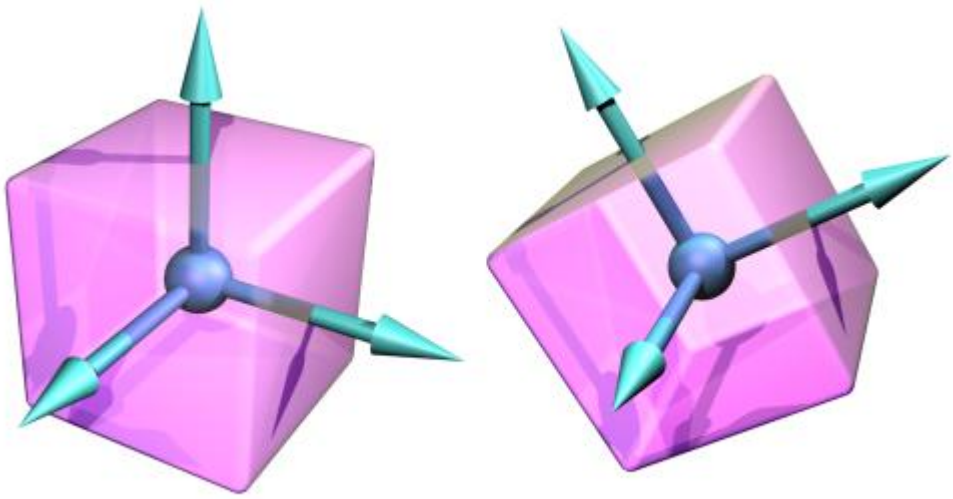
- Биеийн тодорхойлолтын цэгийн байрлалыг заах вектор. (x,y,z) – тухайн хугацаанд дахь биеийн тодорхойлолтын цэгийн байрлал.

- Биеийн тодорхойлолтын цэгийн шугаман хурдны вектор. (vx,vy,vz) – тодорхой хугацааны дараа биеийн тодорхойлолтын цэг ямар зайд шилжихийг заана.
- Биеийн эргэлтийн байрлалын тодорхойлогч. (qs,qx,qy,qz) эсвэл 3x3 хэмжээтэй матриц – тухайн агшин дахь биеийн эргэлтийн байрлалыг заана.
- Өнцгөн хурдны вектор. (wx,wy,wz) – тодорхой хугацааны дараа бие ямар өнцгөөр эргэлт хийгдэхийг заана.

Зарим шинж чанарууд нь бүхий л хугацааны туршид ерөнхийдөө тогтвортой байдаг.

- Биеийн масс
- Биеийн тодорхойлолтын цэгээс хамаарсан биеийн массын төвийн байрлал
- Инерцийн матриц. 3x3 хэмжээтэй матриц байх ба энэ нь тухайн биеийн масс массын төвөөсөө яаж хувиарлагдахийг тодорхойлно.

Бие бүр өөртэй нь хамт хөдөлж эргэж байдаг координатын системтэй болохыг зурагт харууллаа.



Зураг 1.10 Биеийн өөрийн координатын систем

Энэ координатын систем нь биеийн тодорхойлолтын цэгээс эхтэй. ODE дахь зарим шинж чанаруудын утга нь биеүүдийн координатын системээс хамааралтай байхад зарим нь ерөнхий координатын системээс хамааралтай байдаг.

➢ Нэгтгэлт

Хатуу биеүүдийн системийг цаг хугацаагаар загварчлах процессийг нэгтгэлтийн процесс гэдэг. Энэ процессийн алхам бүр нь тухайн агшнаас өгөгдсөн хугацааны алхамын дараах агшин дахь бүх хатуу биесүүдийн төлөвүүдийг тогтоодог. Ямар ч процессын үр дүнд дараах шалгуурыг харгалзан үздэг.

- Хэр зэрэг алдаагүй вэ? Бодит амьдралд хэр ойрхон тооцоолол хийж байна вэ?
- Хэр найдвартай ажиллах вэ? Тооцооллын алдаа гарахгүй байх.

➢ Хүчний аккумлятор

Нэгтгэлтийн процессийн алхамын хооронд хэрэглэгч хатуу биед ямар нэг хүч өгсөн функцуудыг дуудах боломжтой. Энэ хүчнүүд хатуу биеийн хүчний аккумляторт нэгтгэгддэг. Нэгтгэлтийн дараагийн алхамд өгөгдсөн бүх хүчний нийлбэр биеийг хөдөлгөнө. Хүчний аккумлятор нэгтгэлтийн процессийн алхамын дараа тэглэгднэ.

➢ Харилцан үйлчлэлтэй ажиллах

Хатуу биесийн хоорондын харилцан үйлчлэл болон орчны тогтмол биесүүдтэй харилцан үйлчлэх үйлчлэлтэй дараах маягаар ажиллана.

- Загварчлалын алхам бүрийн өмнө хэрэглэгч юу юутай мөргөлдсөнийг тодруулах харилцан үйлчлэлийг тогтоох функцыг дуудна. Энэ функц нь хүрэлцсэн цэгүүдийн жагсаалтыг буцаана. Шүргэлтийн цэг бүр огторгуй

дахь байрлал, гадаргуун нормал вектор, нэвтрэлтийн гүн зэргийг тодорхойлно.

- Загварчлалын алхам явагдана.

➢ Нийтлэг загварчлалын код

- Динамик орчинг үүсгэнэ
- Динамик орчиндоо биесүүдийг үүсгэнэ
- Бүх биесийн төлөвүүдийг тогтооно
- Хэрэгцээтэй бол харилцан үйлчлэлийн орчин болон харилцан үйлчлэлийн геометр объектуудыг үүсгэнэ
- Давталт
  - Хэрэгцээтэй бол биесүүдэд хүчнүүдийг өгнө
  - Харилцан үйлчлэлийн функцийг дуудна
  - Загварчлалын алхамыг явуулна
- Динамик болон харилцан үйлчлэлийн орчныг устгана

1.4.4.2 Орчин

Орчний объект бол хатуу биесүүдийг агуулагч юм. Өөр өөр орчин дахь биесүүд харилцан үйлчлэлцэхгүй. Орчин дахь бүх биесүүд хугацааны тодорхой нэг л агшинд байна. Ихэнх програмуудад нэг л орчин хангалттай байдаг.

1.4.4.3 Автоматаар идэвхитэй ба идэвхигүй төлөвт шилжүүлэх

Ямарч бие идэвхитэй болон идэвхигүй төлөвт байж болно. Идэвхитэй биесүүд загварчлалд оролцох ба идэвхигүй биес загварчлалын алхамд оролцохгүй буюу тооцоологдохгүй. Шинээр үүсгэсэн бие ямагт идэвхитэй төлөвтэй байдаг.

Идэвхигүй биес процессорын ажиллагаанд оролцохгүй ба тиймээс загварчлах хурдыг ихэсгэхийн тулд тэднийг амарч байх хугацаанд нь идэвхигүй төлөвт байлгах хэрэгтэй. Үүнийг автомат идэвхгүйжүүлэгчийн тусламжтайгаар хийж болно.

Хэрэв биесийн автомат идэвхигүйжүүлэлтийн төлөв нь нээлттэй бол дараах тохиолдлуудад бие автоматаар идэвхигүй төлөвт шилжинэ.

- Тодорхой алхмуудад ажиллагаагүй байх бол
- Тодорхой хугацаанд ажиллагаагүй байх бол

Бие ажиллагаагүй байна гэдэг нь биеийн шугаманд ба өнцгөн хурд нь биеийн өгөгдсөн хөдлөх хурдны хязгаараас даваагүй тохиолдол юм.

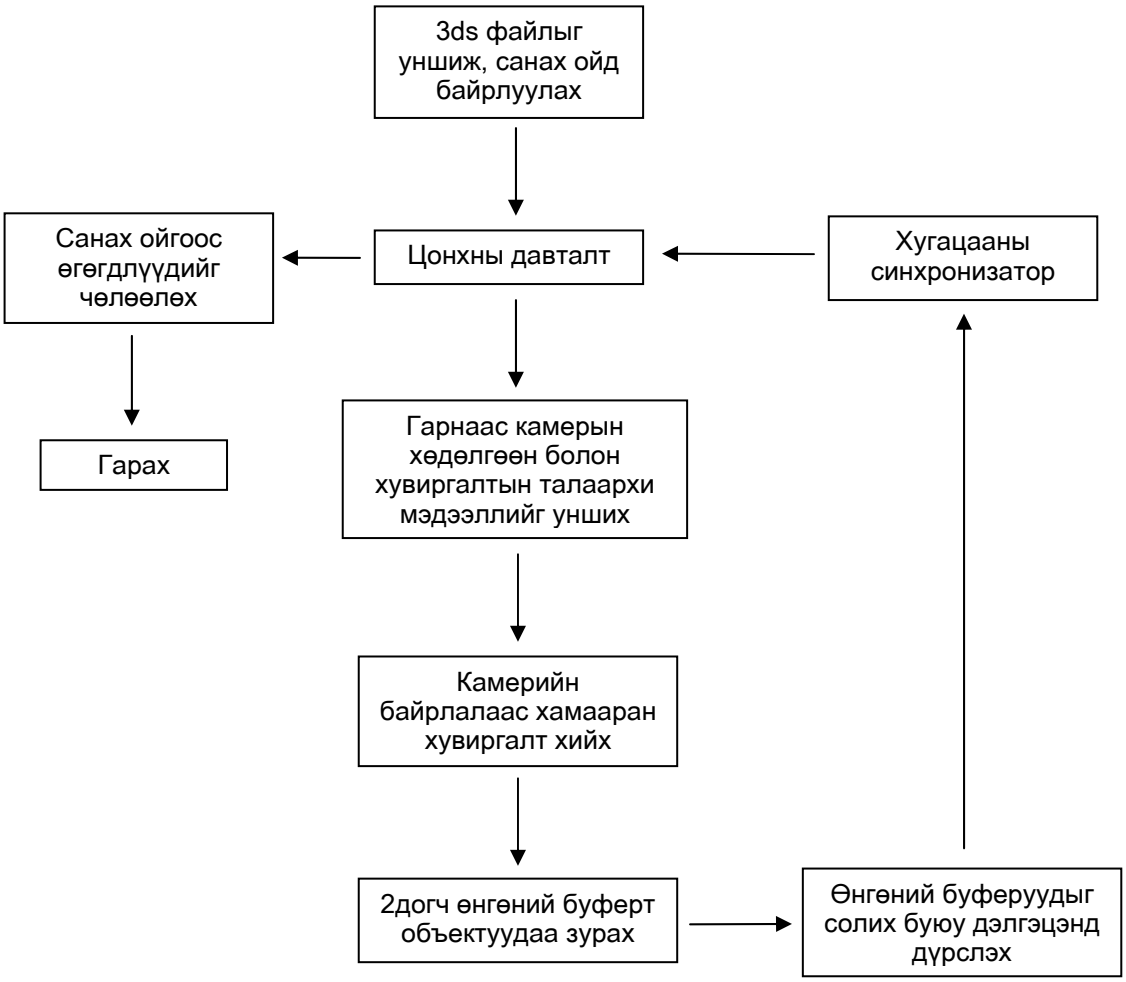
Иймээс бие бүр дараах таван төлөвтэй байна. Идэвхитэй эсэх, ажиллагаагүй байх алхамын тоо хэмжээ, ажиллагаагүй байх хугацааны тоо хэмжээ, болон биеийн шугаман болон өнцгөн хөдлөх хүчний доод хэмжээ. Шинээр үүсгэгдсэн бие эдгээр төлөвүүдээ орчноосоо авдаг. Доорх функцууд эдгээр төлөвүүдийг тогтоох болон авах, идэвхитэй болон идэвхигүй төлөвт шилжүүлэх үүргийг гүйцэтгэдэг.

1.4.5 Хугацааны синхронизатор

Төрөл бүрийн дэлгэцийн карт дээр нэг фрейм буюу нэг кадрыг харилцан адилгүй хугацаанд зурна. Тэгэхлээр удаан компьютер дээр удаан, хурдтай компьютер дээр хурдан, олон объект харагдвал удаан, цөөхөн объект харагдвал хурдан ажиллана. Яг л 286 компьютерт зориулсан Digger тоглоом шиг. Энэ тоглоомонд хугацааны синхронизатор байхгүй учраас 486 процессор дээр бүх зүйл нь хурдан ажиллаад жирэлзээд л өнгөрдөг. Хугацааны синхронизатор нь зурах фреймд яг тухайн хугацааны агшинд зурагдах ёстой дүрс байхаар тохируулдаг. Зурж амжихгүй байгаа зүйлсээ зурахгүй гэсэн дүрэмтэй ажилладаг. Өөрөөр хэлбэл секундэд 60 кадр зурагдах ёстой байхад дэлгэцийн кард 25 кадр л зурж байхад тоглоом ажиллах боломжгүй болохгүй, зөвхөн ойролцоогоор 2.4 кадрыг л алгасаж яг тухайн хугацааны агшинд зурах ёстой зүйлсээ л зурна. Ингэснээр жишээ нь тоглоомонд ямар нэг машин 1 секундэд 1 метр газар явдаг бол дэлгэцийн карт хэдэн фрейм зурахаас үл хамааран зурах агшиндаа машины тухайн хугацаан дахь байрлалыг зурна гэсэн үг. 1 секундэд зөвхөн 2 кадр зурдаг компьютер дээр уг машин эхний фрейм дээр 0.5 метр дараагийн кадр дээр дахин 0.5 метр шилжилт хийгдсэнээр зурагдана. Харин 10 кадр зурдаг компьютер дээр тухай бүр нь 0.1 метр газар шилжилт хийгдсэнээр 10 кадр 1 секундэд зурагдана. Ингэснээр хугацааны хувьд гажилт гарахгүй зөвхөн дүрслэлийн хувьд л гацалдалт гарна гэсэн үг. Компьютер процессорынхоо ачааллаас хамаарч тодорхой хугацаанд тогтмол бус тооны фрейм зурдаг. Түүнийг тухай бүр нь тооцоолон зурагдах ёстой агшинг тооцоолохоор зохион байгуулна.

1.4.6 Тоглоомын давталт

Тоглоомыг тоглож байх явцад цонх, гар, хулгана, дэлгэц гэх зэрэг бүхий л зүйлсүүдийг удирддаг том давталт явагдаж байдаг. Үүнийг бүр тусгайлан тоглоомын давталт гэж нэрлэдэг. OpenGL программ бүр ийм давталттай байдаг. Уг программд тоглоомын давталтыг яаж зохион байгуулсаныг диаграммаар үзүүлье.



Зураг 1.11 Тоглоомын давталт

1.5 СИСТЕМИЙН ЗОХИОМЖ

1.5.1 Бүтэц зохион байгуулалт

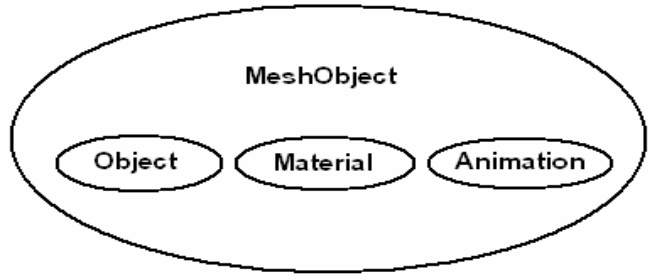
Уг 3 хэмжээст тоглоомын хөдөлгүүр нь Class Library (.DLL файл) болон зохион байгуулагдан тоглоомын програмчлалд ашиглагдана. Өөрөөр хэлбэл дээрх хэрэглэгчийн шаардлагад тусгагдсан бүх зүйлсүүдийг агуулсан классуудын цогц юм. Харин хэрэглэгчид бидний компиляци хийсэн .dll файлыг шууд C# прожектдоо зарлан тоглоомынхоо програмчлалд шууд ашиглана.

Доорх код файлуудыг агуулна.

- 3DSLoader.cs
- BLALoader.cs
- ImageLoader.cs
- VideoPlayer.cs
- DirectPlayClient.cs
- DirectPlayServer.cs
- Structs.cs
- 3Dmath.cs
- Font.cs
- Camera.cs
- KeybInput.cs
- MouseInput.cs

❖ 3DSLoader.cs

MeshObject, Object, Material, Animation зэрэг классуудаас тогтоно. Object, Material, Animation классууд нь MeshObject класст хэрэглэгдэнэ. /Зураг 1.12/



Зураг 1.12 3DSLoader.cs Бүрдмэл харьцаа

MeshObject классын функцуудыг доор үзүүлэв.

```
public MeshObject(String FileName);
```
Файлыг нээж бүтцийг уншин санах ойд хадгалах байгуулагч функц. FileName параметрт .3ds файлын нэрийг өгнө.

```
public void Render();
```
OpenGL график орчинд дүрслэн үзүүлэх функц.

```
public Vector3 CheckCollision(Vector3 pos,float radius);
```
Объектыг *pos* байрлал дахь *radius* радиустай бөмбөрцөгтэй огтлолцсон эсэхийг шалган огтлолцсон бол объектоос цэгийн координатыг *radius* зайд холдуулсан координатыг буцаах функц. Хэрэв огтлолцоогүй бол *pos* –г шууд буцаана.

.3ds файл нь мод бүтэцтэй учраас түүнийг доорх шатласан процедур маягаар уншина.

```
void Load3DS();                          – файлыг таних
└► ulong ReadMain();
    ├► ulong ReadVersion();              – файлын хувилбарыг унших
    ├► ulong ReadMesh();                 – тор бүтцийг унших
    │   ├► ulong MeshVersion();          – тор бүтцийн хувилбар
    │   ├► ulong ReadMasterScale();      – ерөнхий хэмжүүр
    │   ├► ulong ReadObj();              – объектууд
    │   │   └► ulong TriangularMesh();   – гурвалжин тор бүтэц
    │   │       ├► ulong VertexList();   – цэгүүдийн жагсаалт
    │   │       ├► ulong FacesDesc();    – гадаргуун тодорхойлогчид
    │   │       │   ├► ulong FacesMat(); – гадаргуун материал
    │   │       │   └► ulong SmoothGroup();
    │   │       ├► ulong MapCrdList();   – текстур координат
    │   │       └► ulong MeshMatrix();   – тор бүтцийн хув./матриц
    │   └► ulong Material();             – материалууд
    │       ├► ulong MaterialBlock();    – материалын нэр
    │       ├► ulong MatAmbient();       – амбиентийн гэрлийн өнгө
    │       ├► ulong MatDiffuse();       – ойлтын гэрлийн өнгө
    │       ├► ulong MatSpecular();      – сарних гэрлийн өнгө
    │       ├► ulong MatShininess();     – өөрийн гэрэлтэлтийн өнгө
    │       └► ulong MatTransparency();  – нэвт харагдалтын хувь
    └► ulong KeyFrame();
```

Бүх процедурууд файлаас уншсан хэмжээгээ байтаар буцаана. Ингэснээр эцэг процедур нь өөрийн унших хэмжээгээ тодорхойлох боломжтой болно. Энэ хэмжээ нь давталтын төгсгөл болох ёстой.

❖ BLALoader.cs

3 хэмжээст моделийн хөдөлгөөнийг .bla өргөтгөлтэй зохиомол төрөлт файлаас уншин дэлгэцэнд зурах, хөдөлгөөнийг дүрслэх ModelAnimation классыг агуулна.

```
public ModelAnimation(String fname)
```

Файлыг нээж бүтцийг уншин санах ойд хадгалах байгуулагч функц.
fname параметрт .bla файлын нэрийг өгнө.

```
public void Render()
```

OpenGL график орчинд дүрслэн үзүүлэх функц.

```
public void Animate(int start,int end,float fragment)
```

OpenGL график орчинд хөдөлгөөнийг дүрслэн үзүүлэх функц. start параметрт хөдөлгөөний фреймийн эхлэлийн индекс, end параметрт төгсгөлийн индекс, fragment параметрт фрейм хоорондын хугацааны агшинг 0-1 хүртэлх бутархай тоогоор өгнө. Жишээ нь 30 кадртай .bla файл байсан ба түүний эхний 10 кадр дахь хөдөлгөөний хугацааны 1/3 дахь агшинг дүрслэн үзүүлье гэвэл Animate(0,9,0.333333) гэх маягаар параметрүүдийг олгоно. Уг функц нь моделийн хөдөлгөөний хоёр кадрын хоорондох хугацааны агшин дахь моделийн хөдөлгөөнийг ойролцоогоор (пропорционалиар) боддог. Ийм учраас хугацааны фрагментийг бутархай тоогоор оруулж өгсөн.

▪ .bla файлын бүтэц

.bla файлын төрөл нь зохиомол төрөл бөгөөд үүнийг дараах бүтэцтэйгээр зохион байгууллаа.

```
Char[4] id;                 // .bla файлыг таних 4 тэмдэгт ба
                               id == “BLUR” байх ёстой
Int VertexCount;            // Объект дах цэгийн тоо
Int FacesCount;             // Нийт хавтгайн тоо
```

```
Int FramesCount;            // Фреймын тоо
Char[] fname;               // Текстур файлын нэр
                              (NULL terminated string)

Struct {                    // Цэг бүр дээрх текстурын U,V координат
Float X,Y;
} MapCrdList[VertexCount];

Struct {                    // Хавтгайн тодорхойлогч
Int V1,V2,V3;
} Faces[FacesCount];

Struct {                    // Фрейм бүрт харгалзах цэг бүрийн
Float X,Y,Z;                //  координатууд
} Vertices[VertexCount][FramesCount];
```

- ImageLoader.cs

ImgLoader классаас тогтох бөгөөд уг класс нь .bmp, .tga төрөлтэй файлыг уншин санах ойд OpenGL текстур болгон байрлуулна. Нэвт харагдалтгүй зургийг .bmp, нэвт харагдалттай зургийг .tga файлын төрлөөр уншина.

```
public static uint BitmapLoad(String fname,bool clamp)
```

.BMP файлыг уншин OpenGL текстур болгон санах ойд хадгалах функц. fname-д файлын нэрийг өгнө. clamp булын хувьсагчийн утга үнэн бол зураг давтагдалтгүй, худал бол давтагдалттайгаар зурагдах болно. Тэнгэр, өвс зэргийг зурахад ашиглагдана. Өвс давтагдалттай зурагддаг, харин тэнгэр давтагдалтгүй, зааггүй зурагдах ёстой.

```
public static uint LoadTGA(String fname)
```

.BMP файлыг уншин нэвт харагдалттай OpenGL текстур болгон санах ойд хадгалах функц. Мод, өвс зэргийг зурахад ашиглана. fname-д файлын нэрийг өгнө.

- VideoPlayer.cs

Видео өгөгдлийг уншин шууд дэлгэцэнд дүрслэх VideoPlayer классаас тогтоно. Видео өгөгдөл нь эхлэлийн логог харуулах, реклам үзүүлэх зэрэгт

хэрэглэгдэнэ. ActiveX component ашиглаж гарагж байгаа учраас бүхий л төрлийн кодлосон, кодлоогүй видео өгөгдлүүдийг унших боломжтой.

```
public VideoPlayer(String fname)
```

fname нэртэй файлыг уншин дэлгэцэнд үзүүлнэ. Дуусмагц өөрөө хаагдан устана.

- Structs.cs

3 хэмжээст графикт хэрэглэгдэх өгөгдлийн бүтцүүдийг агуулна.

- 3Dmath.cs

Тоглоомын физикт хэрэглэгдэх 3d хэмжээст математик, геометрийн томъёонуудыг агуулсан Math3D классыг агуулна.

Math3D классын гишүүн функцууд:

```
public static float PointToPlaneDistance(Vector3 point,
       Vector3 normal,Vector3 pointonplane)
```

Цэгээс хавтгай хүртэлх зайг буцаах функц.

```
public static bool IsPointOnThePolygon(Vector3 point,
       Vector3 vert1,Vector3 vert2,Vector3 vert3)
```

Тухайн цэг олон өнцөгт дээр оршиж байгаа эсэхийг шалгах функц.

```
public static float AngleBetweenVectors(Vector3 vec1,
       Vector3 vec2)
```

Хоёр векторын хоорондох өнцгийг бодох функц.

- Camera.cs

Камерийн байрлалыг агуулсан камерийн бүх хөдөлгөөнүүд, хүндийн хүч, хугацааны синхронизерийг агуулсан функцуудтай Camera классыг агуулна.

Camera классын өгөгдлүүд

```
public Vector3 pos;    - камерийн байрлал
public Vector3 view;   - камерийн харах вектор
public Vector3 upvec;  - камерийн толгойн харах вектор
public Vector3 foot;   - камерийн өндөр

private float movespeed;    - явах хурд
private float fallspeed;    - унах хурд
private float jumpspeed;    - үсрэх хүч
private float gravity;      - хүндийн хүч


private float currentRotY;  - камерийн харах зүгийн Y тэнхлэгтэй
                              үүсгэх өнцөг
private float mouseSens;    - хулганы эргэх хурд
private float lasttime;     - сүүлийн фрейм дэх хугацаа
public float elapsedtime;   - сүүлийн фреймээс хойшхи хугацаа
private float fps;          - нэг секундэд дэлгэцэнд дүрслэж буй
                              фреймийн тоо frames per second
```

```
public Camera(float posx,float posy,float posz,
           float viewx,float viewy,float viewz,
           float upvecx,float upvecy, float upvecz);
```

Камерийн байгуулагч функц. Байрлал, харах чиг, толгойн векторийг параметрээр авна.

```
public void SetViewByMouse(Vector3 mousePos);
```

Хулганын хөдөлгөөнөөр харах чигийг өөрчлөх функц

```
private void RotateView(float angle,float x,float y,float z);
```

Камерийн харах чигийг өнцгөөр өөрчлөх функц

```
public void MoveCamera(bool forward,bool backward,
                bool left,bool right,bool walk);
```

Камерийг хөдөлгөх функц

```
private void MoveForward(float k);
```

Чигээрээ явах

```
private void MoveBackward(float k);
```

Хойшоо явах

```
private void StrafeLeft(float k);
```
Зүүн тал урууган явах

```
private void StrafeRight(float k);
```
Баруун тал урууган явах

```
public void Duck();
```
Суух

```
public void Stay();
```
Босох

```
public void Jump();
```
Үсрэх

```
public void Fall();
```
Хүндийн хүчээр доош унах, уналтийн хурдыг хугацаагаар тооцоолох

```
public void RePosition(Vector3 pos1);
```
Камерийг зөөх. Хананд тулах зэрэг камерийн хөдөлгөөний дараа камерийн байрлалд засвар хийхэд ашиглана.

```
public void CalculateFrameRate();
```
Сүүлийн фрейм зурагдсанаас хойш өнгөрсөн хугацааг тооцоолох

```
public void Look();
```
Камераар орчинг дүрслэх OpenGL функц.

- KeybInput.cs

DirectX –н DirectInput ашиглан гарны товч дарагдсан төлөвийг унших KeybInput классыг агуулна. Гарны товч бүрт дарагдсан, дарагдаагүй, яг тухайн агшинд дарагдаж байгаа, тухайн агшинд дарагдаагүй төлөвт орж байгаа гэсэн 4 төлөвийг тодорхойлох чадвартай.

```
public void Read();
```
Гарны төлвийг унших функц

```
public bool KeyPressed(Key k);
```
Тухайн товч дарагдсан эсэхийг шалгах функц

```
public bool KeyNotPressed(Key k);
```
Тухайн товч дарагдаагүйг шалгах функц

```
public bool KeyDown(Key k);
```
Товч яг тухайн агшинд дарагдсан төлөвт орж байвал үнэн утга буцаана.

```
public bool KeyUp(Key k);
```
Товч яг тухайн агшинд дарагдаагүй төлөвт орж байвал үнэн утга буцаана.

- MouseInput.cs

DirectX –н DirectInput ашиглан хулганы товчнуудын төлөв болон хулганы зөөлтийн векторыг унших MouseInput классыг агуулна. Хулганы товч бүрт дарагдсан, дарагдаагүй, яг тухайн агшинд дарагдаж байгаа, тухайн агшинд дарагдаагүй төлөвт орж байгаа гэсэн 4 төлөвийг тодорхойлох чадвартай.

```
public void Read();
```
Хулганы төлвийг унших функц

```
public bool ButtonPressed(Key k);
```
Тухайн товч дарагдсан эсэхийг шалгах функц

```
public bool ButtonNotPressed(Key k);
```
Тухайн товч дарагдаагүйг шалгах функц

```
public bool ButtonDown(Key k);
```
Товч яг тухайн агшинд дарагдсан төлөвт орж байвал үнэн утга буцаана.

```
public bool ButtonUp(Key k);
```

Товч яг тухайн агшинд дарагдаагүй төлөвт орж байвал үнэн утга буцаана.

```
public Vector3 Move();
```

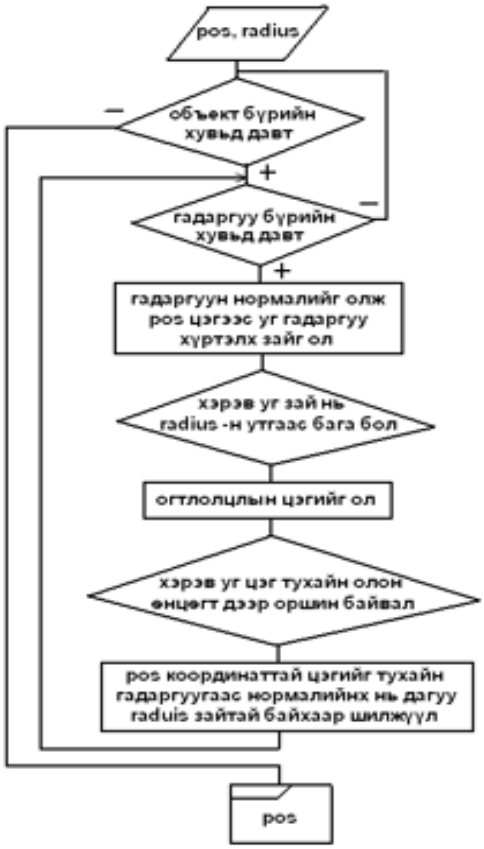
Хулганы зөөлтийн векторыг буцаах функц

1.5.2 Физик хөдөлгүүр

- ❖ Тулалтыг мэдрэх физик хөдөлгүүр

Уг физик нь MeshObject класс дотор CheckCollision функц хэлбэрээр байрлаж байгаа бөгөөд 3Dmath.cs –н математик функцуудыг ашиглана.

Доорхи алгоритмаар дүрслэн үзүүлэв.



Зураг 1.13 Тулах физикийн алгоритм

```
public Vector3 CheckCollision(Vector3 pos,float radius);
```

Объектыг *pos* байрлал дахь *radius* радиустай бөмбөрцөгтэй огтлолцсон эсэхийг шалган огтлолцсон бол объектоос цэгийн координатыг *radius* зайд холдуулсан координатыг буцаах функц. Хэрэв огтлолцоогүй бол *pos* –г шууд буцаана.

- ❖ Хүндийн хүчний физик

Хүндийн хүчний физик алгоритмыг Camera класс дотор бодно.

```
public void Fall();
```

Хүндийн хүчийг тооцоолон уналтыг бодох функц

$$S = V_0 * t + \frac{gt^2}{2}$$

$$V_t = V_0 + g * t$$

1.5.3 Хугацааны синхронизатор

Манай 3D Game Engine –н хувьд хугацааны синхронизатор Camera класс дотор хийгдсэн. Камерийн хөдөлгөөн, моделийн хөдөлгөөн бүр хугацаатай шууд хамааралтайгаар явагдах ёстой.

```
public void CalculateFrameRate();
```

Өмнөх фреймээс хойш өнгөрсөн хугацааг тооцоолох.

1.5.4 BLA File Creator хөдөлгөөний файл үүсгэгч програм

BLA File Creator програм нь 3DSLoader класстай төстэй зохион байгуулалттай, 3ds файлыг уншин хэрэгтэй мэдээллүүдээ аваад өөрсдийн зохиосон .bla төрөлт файлыг үүсгэдэг DOS орчны програм юм.

Бид 3 хэмжээст моделио 3D Studio Max програмаар зурсны дараа түүний хөдөлгөөнийг кадр бүрээр нь экспортлож олон тооны 3ds файлыг үүсгэнэ. Бүх 3ds файлууд ижил нэрээр эхлэсэн байх ба түүний араас 0-с эхлэсэн кадрын индекс тоо байна. Уг файлуудыг BLA File Creator програмаар нэгтгэн .bla файл гарган авна.

1.5.5 Өгөгдлийн бүтэц

3ds файлд цэгийг x,y,z координатуудаар, хавтгайг 3 цэгээр буюу энгийн хавтгайн тодорхойлогчоор, объектуудыг 3 өнцөгт хавтгайнуудын бүрдэлээр, өнгийг улаан, ногоон, цэнхэр өнгүүдээр, векторыг x, y, z тэнхлэгийн модулиар дүрсэлнэ.

- Цэгийн бүтэц

```
struct vertex
{ float x;                  // X тэнхлэгийн координат
  float y;                  // Y тэнхлэгийн координат
  float z;                  // Z тэнхлэгийн координат
};
```

- Хавтгайн бүтэц

```
struct face
{ unsigned int vertex1;     // 1-р цэгийн индекс
  unsigned int vertex2;     // 2-р цэгийн индекс
  unsigned int vertex3;     // 3-р цэгийн индекс
  unsigned int mat;         // Хавтгайн материалын индекс
};
```

- Өнгийн бүтэц

```
struct colorf
{ float r;              // Улаан өнгөний эрчимжилт
  float g;              // Ногоон өнгөний эрчимжилт
  float b;              // Цэнхэр өнгөний эрчимжилт
};
```

- Вектор бүтэц

```
struct vector3d
{ float x;
  float y;
  float z; };
```

- Объектын бүтэц

```
class object
{ public:
    char name[30];                  // Объектын нэр
    float mesh_matrix[4][4];        // Хувиргалтын матриц
    vector<vertex>vertices;         // Цэгүүдийн координатууд
    vector<vert2d>mapcrdlist;       // Тухайн цэг бүрт харгалзах
                                       текстур файлын координат
    vector<face>faces;              // Гадаргуугын тодорхойлогчид
    vector<int>smoothlist;          // Гөлгөржилтийн коэф.
};
```

Объект дахь цэгийн тоо, объектын материалын цэгийн координат, гадаргуунууд, гөлгөрын жагсаалт зэргийн тоо нь тодорхойгүй тулд хувьсах урттай массив Microsoft Foundation Classes -н vector class -г авч ашиглав.

- Материал бүтэц

```
class material
{ public:
    char name[30];                  // Материалын нэр
    colorf ambient;                 // Амбиент өнгө
    colorf diffuse;                 // Цацрах өнгө
    colorf specular;                // Үзэгдэх өнгө
    colorf emission;                // Гаргах өнгө
    float shininess;
    float transparency;             // Нэвт харагдалт
    int texmapp;                    // Текстурын процент
    bool hasmap;                    // Тескстуртэй эсэх


    char mapname[30];               // Текстур файлын нэр
    int matindex;                   // Текстур файлын санах ойд
                                       хадгалагдсан индекс
};
```

- Текстурын жагсаалт

```
GLuint texture[256];                // Текстурын санах ойн хаяг
```

- .3ds mesh бүтэц

```
class MeshObject
{ private:
   FILE *Mesh_File;                 // Файлын заагч
   unsigned int File_Version;       // Файлын хувилбар
   unsigned int Mesh_Version;       // Мешийн хувилбар
   int NumberOfObjects;             // Объектын тоо
   int NumberOfMaterials;           // Материалын тоо
   float MasterScale;               // Мешын ерөнхий суналт

   vector<object>objects;           // Объектууд
   vector<material>materials;       // Материалууд
};
```

- Файлын заагч

```
FILE *Mesh_File;
```

1.5.6 Бүтэцлэгдсэн хэл

➢ First Person Shooter төрлийн тоглоомын ерөнхий загвар

- График орчинг бүрдүүлэх (OpenGL цонхыг үүсгэх)
  - Эхлэл лого видеог гаргах
  - Оролт гаралтын объектыг үүсгэх
  - Тоглоомын менюг үүсгэх
    - Зургийн файлуудыг санах ойд байрлуулах
  - Камерын объектыг үүсгэх
  - Моделиудын объектуудыг үүсгэх
    - 3ds файлыг уншин бүтцийг гаргаж аван санах ойд хадгалах
    - Текстур зургийн файлуудыг уншиж санах ойд байрлуулах
  - Сүлжээний интерфейс үүсгэх
  - Тоглоомын давталт
    - Дэлгэцийг цэвэрлэх

- Тоглоомын меню харагдах хэрэгтэй бол зурах
- Оролтын мэдээллүүдийг унших
- Сүлжээгээр ирсэн мэдээллүүдийг унших
- Оролт болон сүлжээний мэдээллээр боловсруулалт хийх
- Физикийн хуулиудаар тооцоололт хийх
- Камерийг байрлуулах
- Санах ойгоос объектын байрлал, хэлбэр дүрс, материал, хөдөлгөөн зэргийг уншин OpenGL функцуудыг ашиглан дэлгэцэнд харуулах

- Санах ойг чөлөөлөх

➢ Зурах функц

Зурах функц бол уг программын хамгийн чухал функц. Санах ойд байрлах объектуудыг тухайн камерын байрлалаар хувиргасан хугацааны нэг агшин дахь зургийг зурна. Зурах функцуудыг бичсэнээр 2догч өнгөний буферт зургийг зурна. Харин буферуудыг солисоноор дэлгэцэнд зурагдана. Зурах алгоритмыг pseudocode -р дүрсэлье.

```
Өнгөний буфер болон гүний буферийг цэвэрлэ;
Камерийн байрлалаар хувиргалт хий;
Давт(Бүх объект бүрийн хувьд)
{
  Хувиргалтын матрицыг стект хий;
  Давт(Бүх гадаргуу бүрийн хувьд)
  {
    Хэрэв(Өмнө зурж байсан материал биш бол)
    {
      Өнгийг одоо зурах өнгөөр соль;
      Материалыг одоо зурах материалиар соль;
    }
    Дүрс зурах(Гурвалжин)
    {
      1-р цэгт харгалзах материалын координат;
```

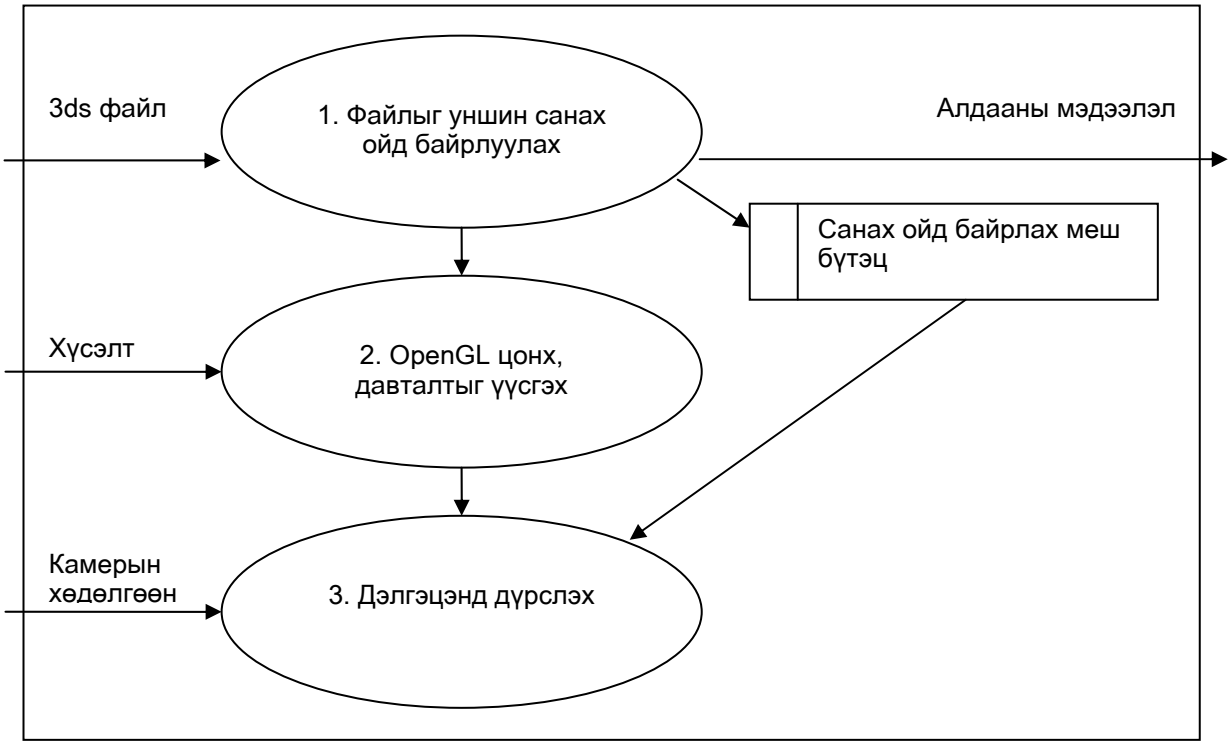
```
      1-р цэгийн координат;
      2-р цэгт харгалзах материалын координат;
      2-р цэгийн координат;
      3-р цэгт харгалзах материалын координат;
      3-р цэгийн координат;
    }
  }
  Хувиргалтын матрицыг стекээс гарга;
}
Өнгөний буферуудыг соль;
```

- BLA File Creator програмын pseudocode

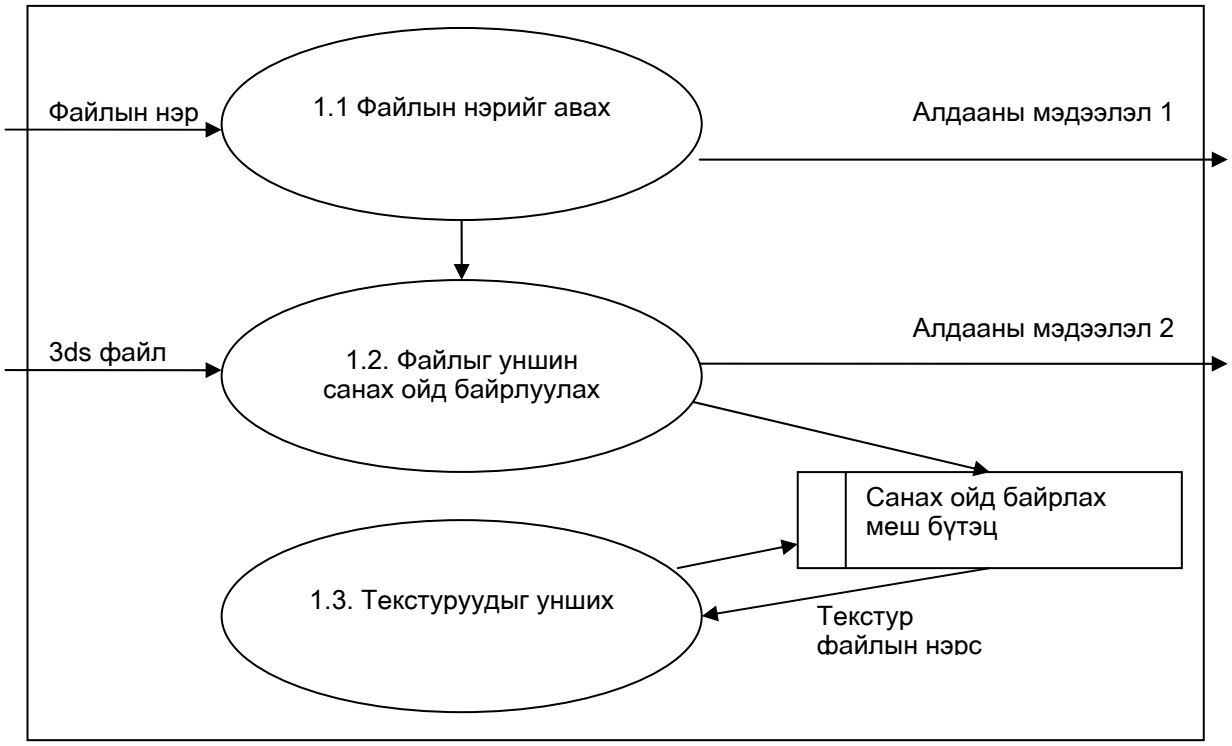
```
Гараас үүсгэх файлынхаа нэрийг авах;
.bla файлыг үүсгэх;
Гараас .3ds файлын эхлэл нэрийг авах;
Гараас нийт кадрын тоог авах;
Объект дахь цэг, гадаргуу, фреймын тоог .bla файлд бичих;
Цэг бүрийн u,v координатуудыг .bla файлд бичих;
Гадаргуугын тодорхойлогчдыг .bla файлд бичих;
Файл буюу фрейм бүрийн хувьд
{
    3ds файлыг нээж унших;
    Файл олдохгүй бол алдааг мэдээлэх;
    .bla файлд фрейм бүрийн хувьд цэг бүрийг координатуудыг
бичих;
}
Файлыг хаах;
```

1.5.7 Өгөгдлийн урсгалын диаграмм

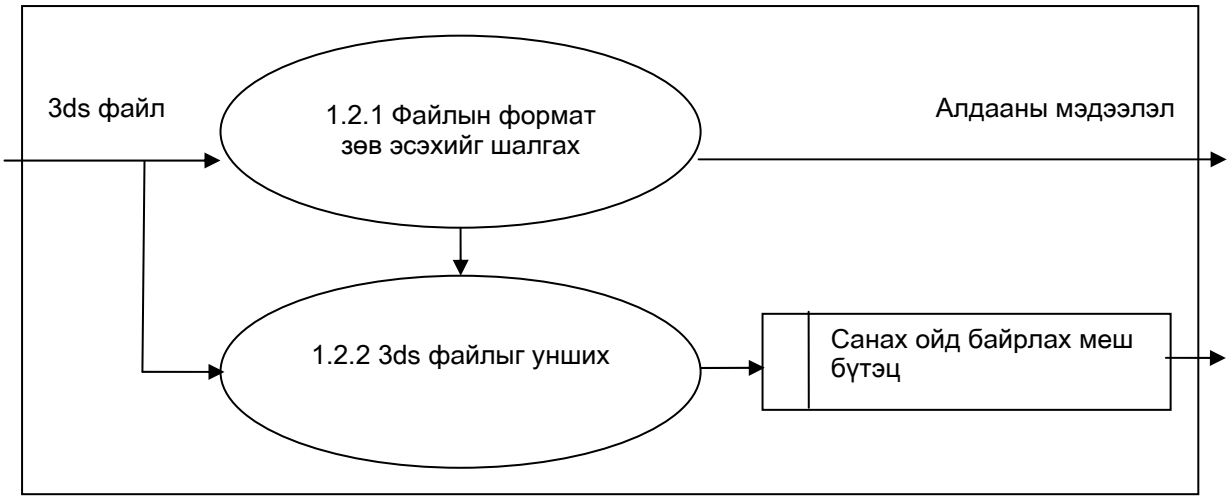
Диаграмм 1, түвшин 0



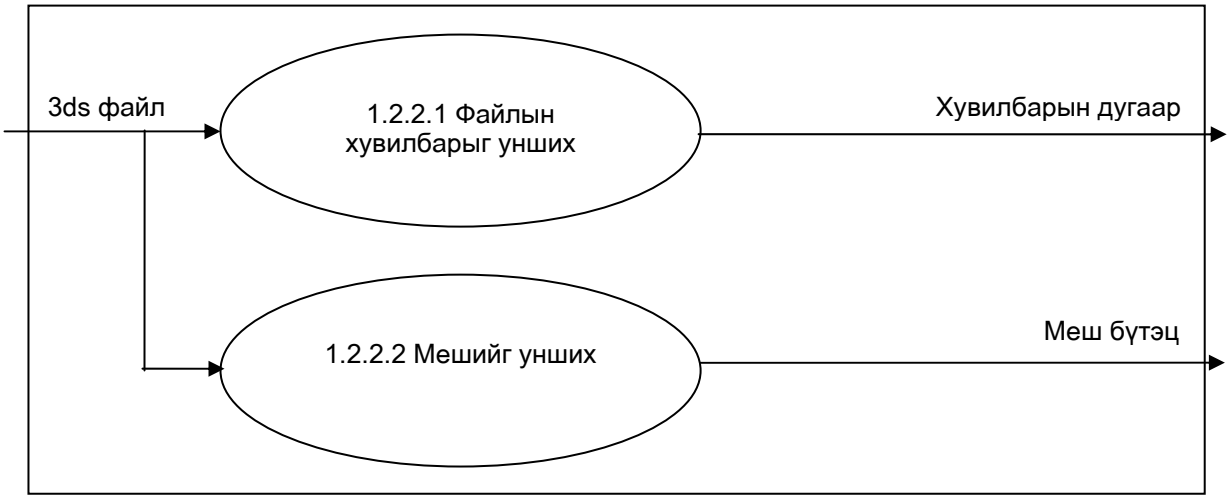
Диаграмм 2, түвшин 1



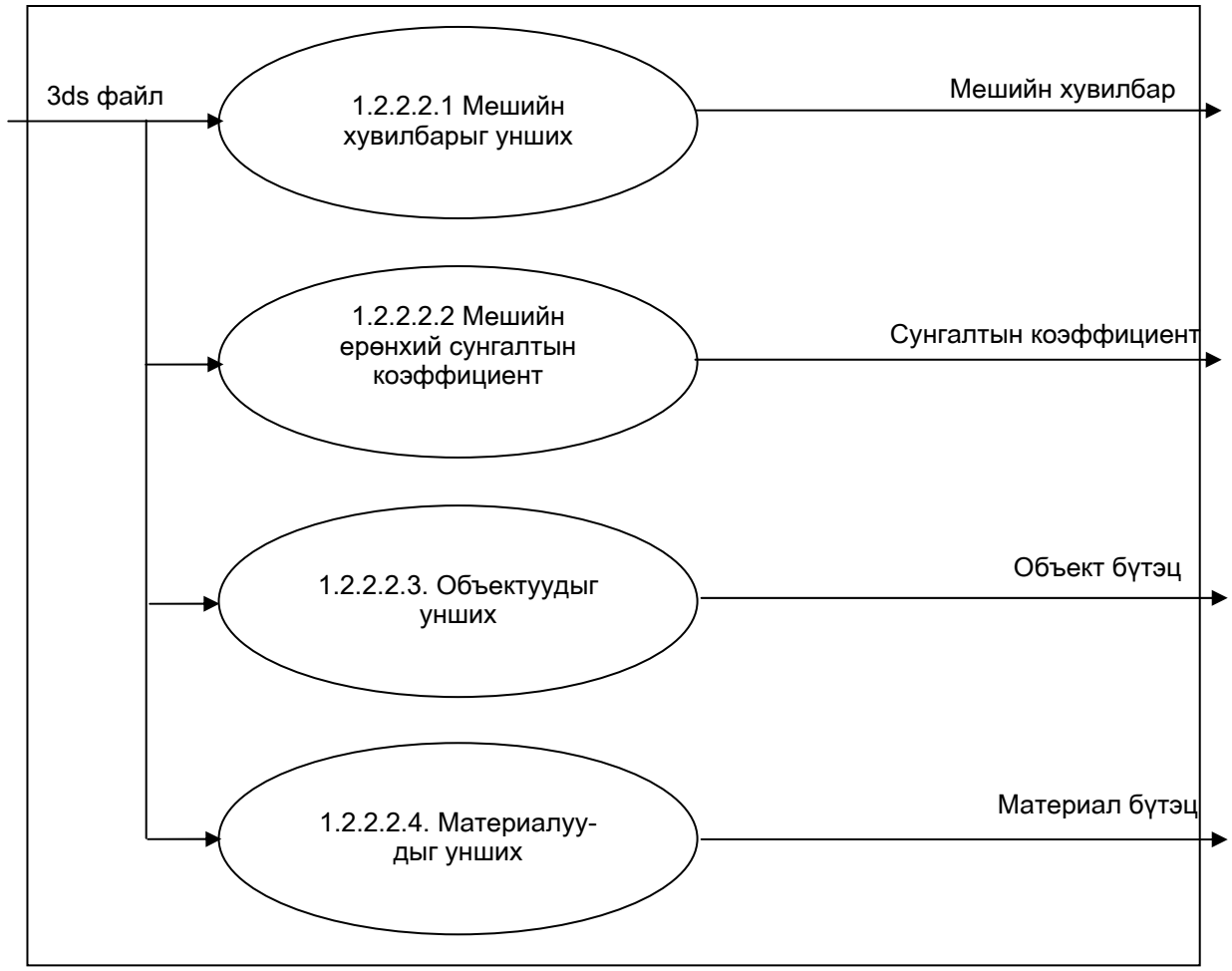
Диаграмм 3, түвшин 2



Диаграмм 4, түвшин 3

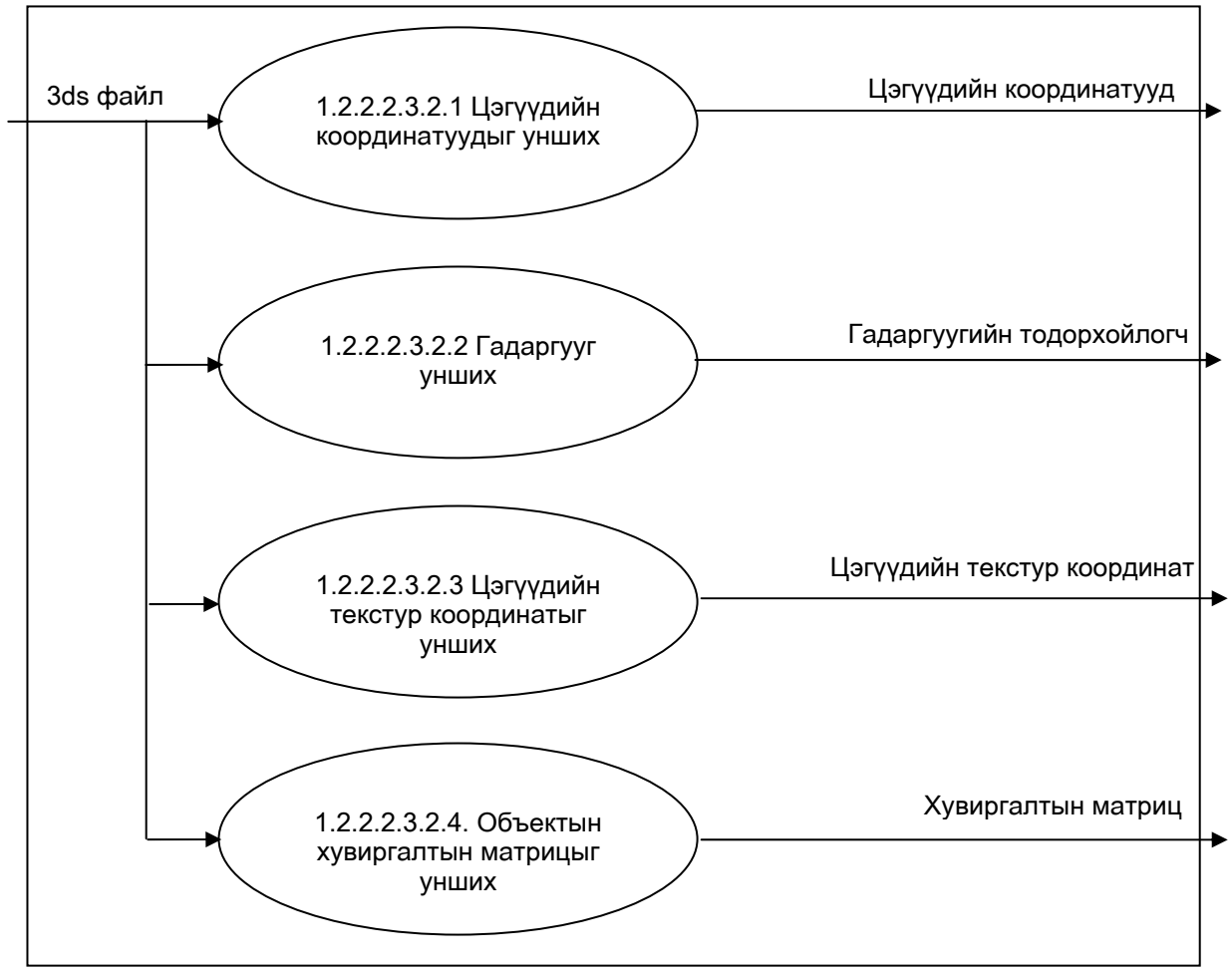


Диаграмм 5, түвшин 4

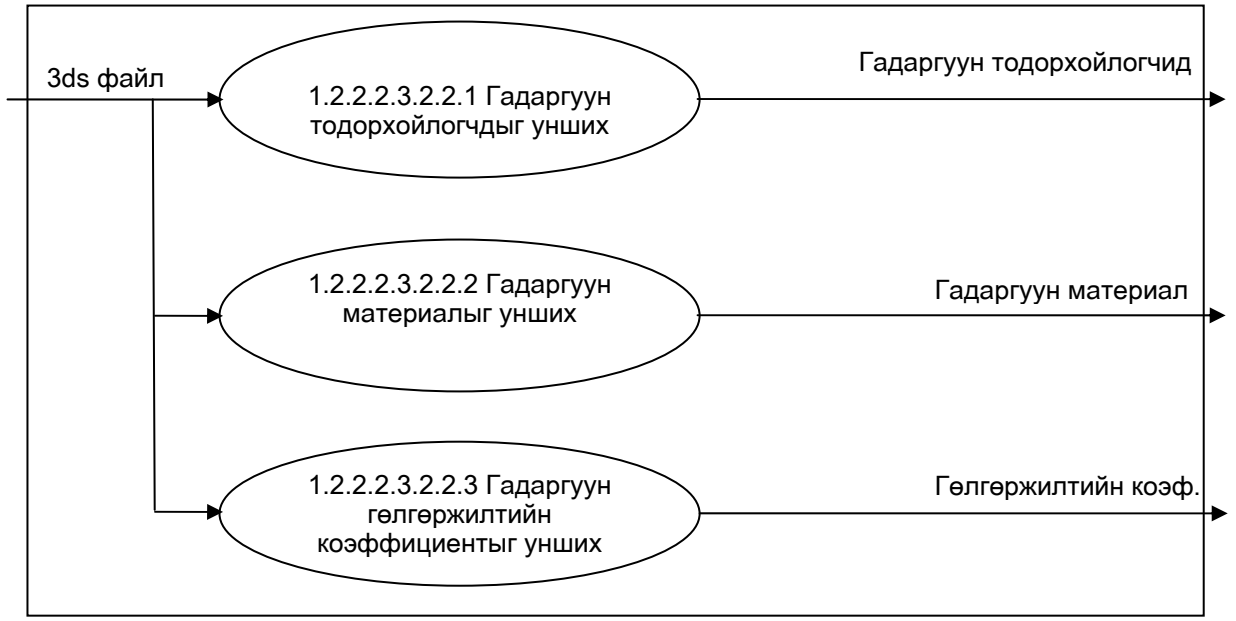


Диаграмм 6, түвшин 5

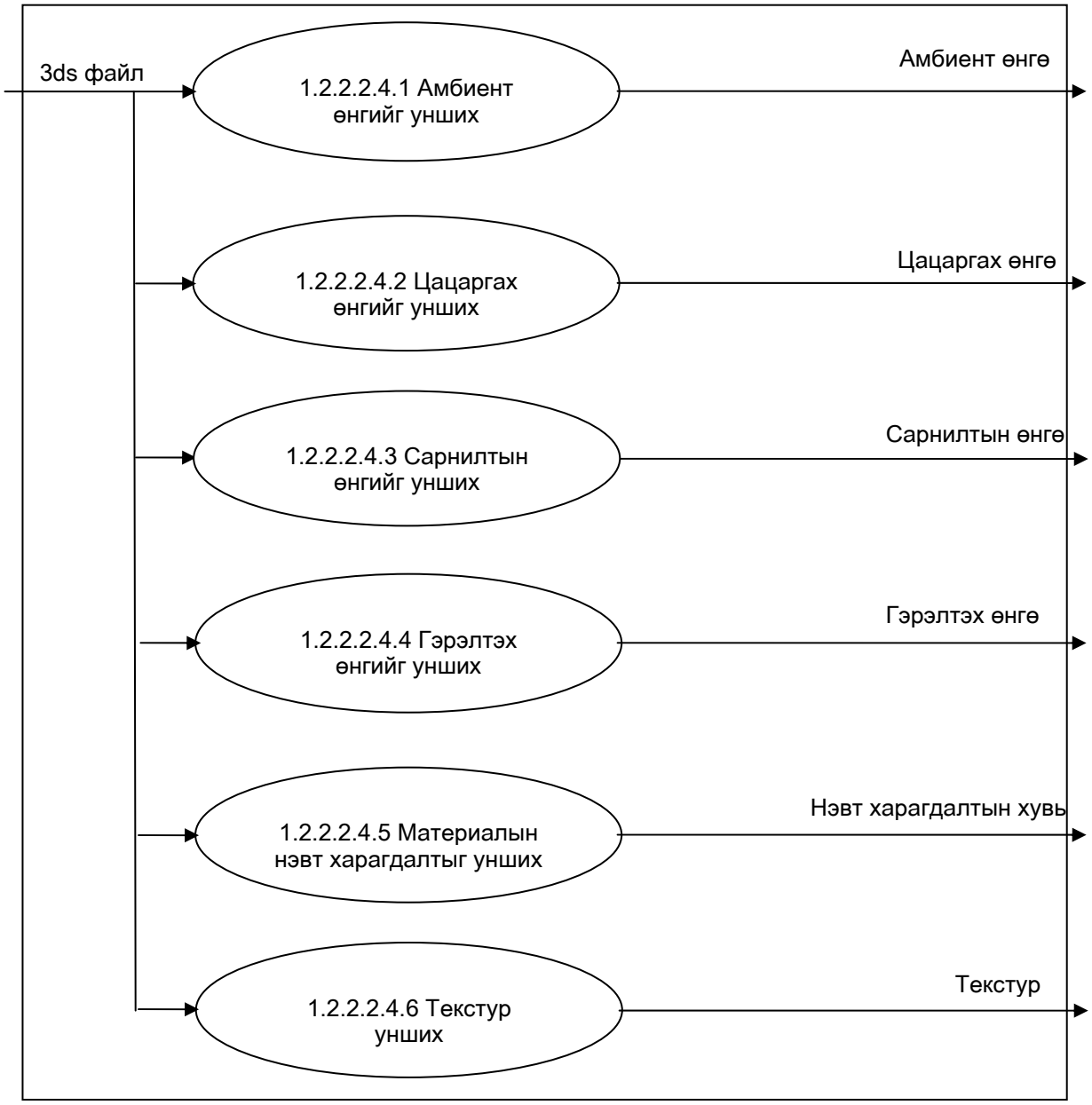
Диаграмм 7, түвшин 6



Диаграмм 8, түвшин 7

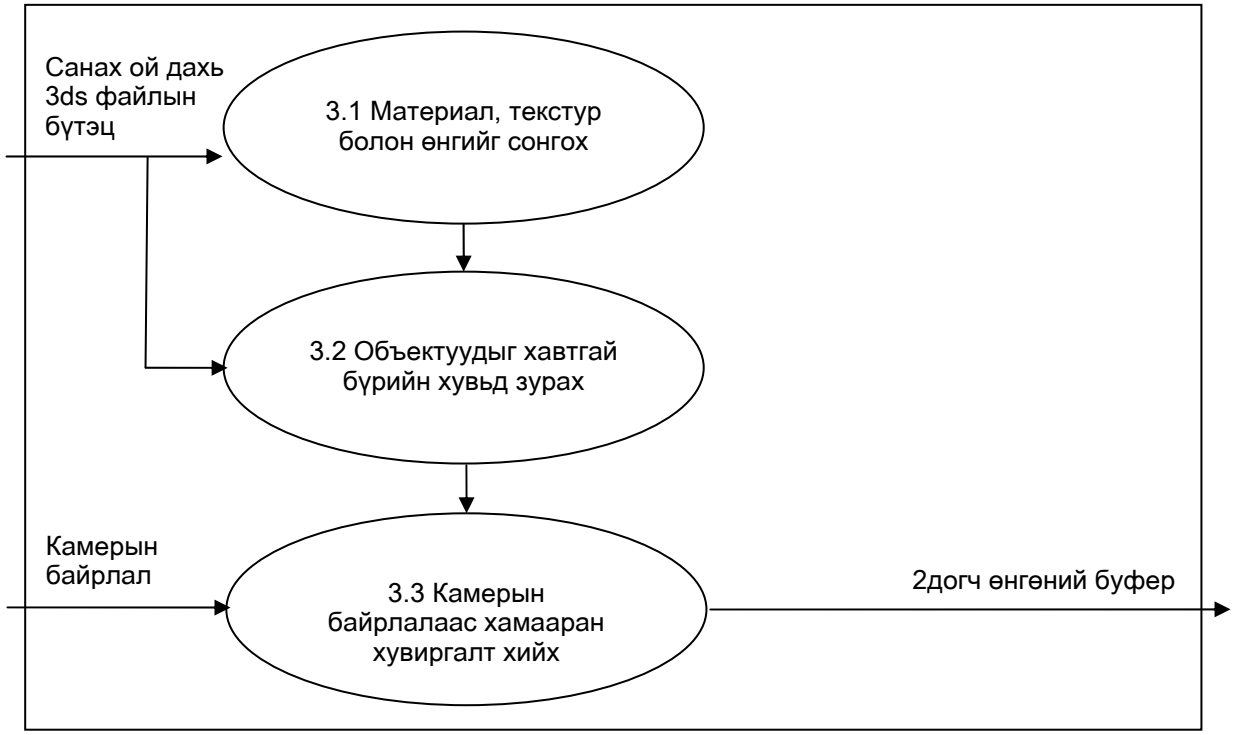


Диаграмм 9, түвшин 5



Диаграмм 10, түвшин 6

Диаграмм 11, түвшин 1



II БҮЛЭГ. ХЭРЭГЛЭГЧИЙН ГАРЫН АВЛАГА

2.1 Танилцуулга

BlurzEngine 3 хэмжээст тоглоомын хөдөлгүүр нь Microsoft .NET платформ, C# хэл дээр Microsoft DirectX өгөгдлийн бүтэц, OpenGL график орчин, CsGL managed code library ашиглан бичигдсэн. Иймээс уг хөдөлгүүрийг ашиглахын тулд дээрх програмчлалын болон график орчнуудыг бүрдүүлж өгөх хэрэгтэй. Үүний тулд доорх зүйлсийг бүрдүүлсэн байх шаардлагатай.

Microsoft .NET Framework
DirectX Managed Code Library
CsGL library (C# OpenGL library)
C# хэл дээр CsGL ашиглан OpenGL орчин мөн тоглоомын давталтыг үүсгэсэн байх шаардлагатай.

Уг 3 хэмжээст тоглоомын хөдөлгүүр нь үүсгэсэн OpenGL орчинд шууд хандан ажиллах болно.

BlurzEngine нь доорх хэсгүүдээс бүрдэнэ.

- 3ds файлыг уншин өгөгдлийг үүсгэж санах ойд ачаалах, дэлгэцэнд дүрслэх **MeshObject** класс
- 3 хэмжээст хөдөлгөөнт .BLA файлыг уншин санах ойд байрлуулах, хөдөлгөөний дурийн агшинг дэлгэцэнд дүрслэх **ModelAnimation** класс
- 2 хэмжээст график буюу .BMP, .TGA файлуудыг санах ойд ачаалах **ImgLoader** класс
- Хүний дотроос харсан камер буюу First Person камерийн бүтэцийг агуулсан, камерын хөдөлгөөнийг удирдах **Camera** класс

- 3 хэмжээст орчин дахь энгийн математик функцууд
- 3 хэмжээст геометр өгөгдлийн бүтэц
- Гравитацын хүч болон биед тулах буюу үл нэвтрэх орчны физик шинж чанарууд
- Компьютерийн гарнаас параллелаар мэдээлэл унших болон боловсруулалт хийх **KeybInput** класс
- Компьютерийн хулганаас мэдээлэл унших болон боловсруулалт хийх **MouseInput** класс

2.2 Програмчлалын болон график орчныг бүрдүүлэх

Microsoft Windows үйлдлийн систем дээр 16MB-с дээш санах ойтой OpenGL-г дэмждэг дэлгэцийн кардыг зөв суулгасан байх (16 болон түүнээс дээш бит өнгөний ялгаралыг дүрслэх чадвартай байх) шаардлагатай.

Microsoft .NET Framework, Microsoft DirectX Managed Code Runtime суулгасан байх шаардлагатай. Мөн CsGL library .DLL файлыг хуулж авсан байх шаардлагатай.

Microsoft C# хэл дээр CsGL ашиглан OpenGL орчин болон тоглоомын давталтыг үүсгэх хэрэгтэй.

2.3 3 хэмжээст биеийг дүрслэх

MeshObject классыг ашиглан 3 хэмжээст биеийг дүрслэдэг тул 3 хэмжээст бие бүрт MeshObject төрөлтэй объект зарлаж өгөх шаардлагатай.

Зарлах

```
Бичигдэх хэлбэр:
MeshObject [объектын нэр];
```

Санах ойд ачаалах

Бичигдэх хэлбэр:

```
[объектын нэр] = new MeshObject("файлын нэр");
```

Файлын нэрд 3 хэмжээст график өгөгдөл буюу .3ds файлынхаа нэрийг зааж өгнө.

Дэлгэцэнд дүрслэх

Бичигдэх хэлбэр:

```
[объектын нэр].Render();
```

Дэлгэцэнд дүрслэхийнхээ өмнө хувиргалтын матрицыг уг биетийн байрлал, эргүүлэлт, сунгалтын матрицаар хувиргасан байх шаардлагатай.

2.4 3 хэмжээст хөдөлгөөнт биеийг дүрслэх

ModelAnimation классыг ашиглан дүрслэдэг.

Зарлах

Бичигдэх хэлбэр:

```
ModelAnimation [объектын нэр];
```

Санах ойд ачаалах

Бичигдэх хэлбэр:

```
[объектын нэр] = new ModelAnimation("файлын нэр");
```

Файлын нэрд 3 хэмжээст хөдөлгөөнт график өгөгдөл буюу .bla файлынхаа нэрийг зааж өгнө.

Дэлгэцэнд дүрслэх

Бичигдэх хэлбэр:

```
[объектын нэр].Animate([эхлэлийн фрейм],
                [төгсгөлийн фрейм],[хөдөлгөөний агшин]);
```

Дэлгэцэнд дүрслэхийнхээ өмнө хувиргалтын матрицыг уг биетийн байрлал, эргүүлэлт, сунгалтын матрицаар

хувиргасан байх шаардлагатай. [эхлэлийн фрейм]-д тухайн хөдөлгөөний .bla файл дахь фреймын дугаарын эхлэлийн дугаарыг оруулна. [төгсгөлийн фреймд]-д тухайн хөдөлгөөний сүүлийн фреймийн дугаарыг оруулна. [хөдөлгөөний агшин]-д эхлэл төгсгөлийг нь тодорхойлсон тухайн хөдөлгөөний хэсгээс ямар агшинг дүрслэхийг тодорхойлсон 0-1 хүртэлх бутархай тоог оруулна.

Жишээ нь: Шувуу объектын 30-45 хүртэлх фрейм нь шувууны газар буух хөдөлгөөнийг дүрслэсэн ба түүнийг буух хөдөлгөөн эхлэсэнээс дуусах хүртэлх хугацааны 1/4 агшин дахь хөдөлгөөнийг зурах хэрэгтэй гэвэл

```
Шувуу.Animate(30,45,0.25);
```

болох юм

2.5 2 хэмжээст график зургийг санах ойд ачаалах

.BMP, .TGA зургийн файлуудыг санах ойд ачаалах

Бичигдэх хэлбэр:

```
ImgLoader.BitmapLoad("файлын нэр",[хүрээтэй эсэх]);
```

.BMP төрлийн файлыг санах ойд ачаалах ба тодорхойлогч дугаарыг буцаах функц

[хүрээтэй эсэх] – залгагдсан зураг эсэхийг тодорхойлох бүлийн хувьсагч

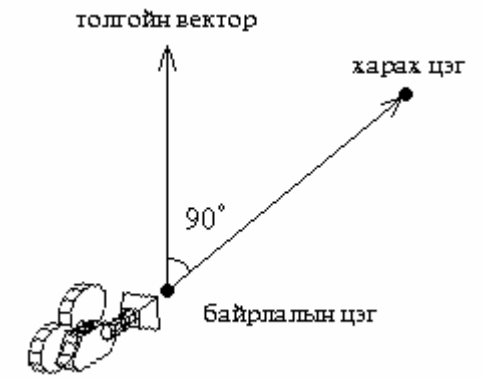
Бичигдэх хэлбэр:

```
ImgLoader.TGALoad("файлын нэр");
```

.TGA төрлийн файлыг санах ойд ачаалах ба тодорхойлогч дугаарыг буцаах функц. 24 болон 32 битийн зургуудыг унших чадвартай. Нэвт харагдалтын alpha layer давхаргатай зураг 32 бит байдаг.

2.6 Камерийн харагдалт болон хөдөлгөөн

Камер нь байрлал, харах чиглэл, толгойн вектор болон камерийн хөдөлгөөний функцуудыг агуулдаг.



Зураг 2.1 Камерийн тодорхойлогч

Камерийг тодорхойлохдоо дээрх 2 цэгийн координат болон толгойн вектороор тодорхойлдог. Байрлалын цэгээс харах цэгрүү харсан, камерийн толгой нь толгойн векторын чиглэлтэй камер байна гэж тодорхойлдог.

Зарлах

```
Бичигдэх хэлбэр:
Camera [объектын нэр];
```

Үүсгэх болон анхны байрлалыг тогтоох

```
Бичигдэх хэлбэр:
[объектын нэр] = new Camera([байрлал x,y,z],
                [харах цэг x,y,z],[ x,y,z]);
```

Харах

```
Бичигдэх хэлбэр:
[объектын нэр].Look();
```

Проекцын матрицд камерын байрлалаар хувиргалт хийх. 3 хэмжээст орчны фреймыг зурах бүрийнхээ өмнө уг хувиргалтыг хийнэ.

Харах

Бичигдэх хэлбэр:

```
[объектын нэр].CalculateFrameRate();
```

Өмнөх зурсан фреймээс зурагдаж буй фрейм хүртэлх хугацааг тооцоолох. Энэ нь биесийн байрлалыг тогтоох хөдөлгөөний агшинг тодорхойлох зэрэгт чухал хэрэгтэй

Хулганы хөдөлгөөнөөр харах чиглэлд эргүүлэлт хийх

Бичигдэх хэлбэр:

```
[объектын нэр].SetViewByMouse([хулганы байрлал]);
```

Хулганыг хөдөлгөх үед камерын харах цэгийг эргүүлэлт хийж өөрчлөх

Камерыг хөдөлгөх

Бичигдэх хэлбэр:

```
[объектын нэр].MoveCamera([урагш],[хойш],[зүүн]
                    ,[баруун],[алхах]);
```

Явах чиглэл болон алхах эсвэл гүйхийг тодорхойлох бүлийн хувьсагчид

Хөдөлгөөний шинж чанарууд

movespeed - хөдөлгөөний хурд

fallspeed - уналтын хурд (тухайн агшин дахь)

jumpspeed – үсрэлтийн хүч

gravity - хүндийн хүч

mouseSens - хулганы хөдлөх мэдрэмж

2.7 3 хэмжээст орчин дахь энгийн математик функцууд

Доорх орчны энгийн 3 хэмжээст математик функцуудыг бичиж өгсөн. (1.4.3-р хэсгээс харна уу)

Цэгээс хавтгай хүртэлх зай

Бичигдэх хэлбэр:

```
PointToPlaneDistance([цэгийн байрлал],[хавтгайн нормал],
                     [хавтгай дээрх дурын цэгийн координат]);
```

Бутархай тоо буцаах функц

Тухайн цэг олон өнцөгт дээр оршиж байгаа эсэхийг шалгах

Бичигдэх хэлбэр:

```
IsPointOnThePolygon([цэгийн байрлал],
                    [олон өнцөгтийн 1,2,3-р цэгийн координат]);
```

Бүлийн утга буцаах функц

Хоёр векторын хоорондох өнцгийг олох

Бичигдэх хэлбэр:

```
AngleBetweenVectors([1-р вектор],[2-р вектор]);
```

Бутархай тоон утга буцаах функц

2.8 3 хэмжээст геометр өгөгдлийн бүтэц

Цэг, хавтгай, вектор, өнгөний өгөгдлийн бүтэц

Төрөл

Vector3D - цэг

Face - хавтгай

Vector3D - вектор

Colorf – өнгө

2.9 Орчны физик шинж чанарууд

- Хүндийн хүчний физик шинж чанар

Хүндийн хүчийг харагдах фрейм бүрт бодно. Биеийн хурд хүндийн хүчний хурдатгалаас хамааран ихэснэ. Өөрөөр хэлбэл бие хурдсан унана. Камерын байрлалд хүндийн хүчний уналтыг оруулж өгсөн.

Уналтын процессийг явуулах

```
Бичигдэх хэлбэр:
[камер объект].Fall();
```

Хурдыг тогтоох

```
Бичигдэх хэлбэр:
[камер объект].fallspeed = [шинэ утга];
```

Хурдатгалыг тогтоох

```
Бичигдэх хэлбэр:
[камер объект].gravity = [шинэ утга];
```

- Биеийн тулалтыг мэдрэх орчны физик

3 хэмжээст биеийг үүсгэж буй класст өгөгдсөн цэг уг биеийн ямар нэг хэсэгт тодорхой хэмжээгээр ойрхон байрласан эсэхийг шалгах, хэрэв ойрхон байрласан бол тодорхой хэмжээний зай авч байрласан цэгийн координатыг буцаах функцыг бичиж өгсөн. Өөрөөр хэлбэл тухайн биед ойртох үед нь мэдэж тодорхой зайд холдож байрлах цэгийг зааж өгдөг гэсэн үг.

3 хэмжээст биеийн хавтгай бүрээс өгөгдсөн цэг хүртэлх зай тодорхой утгаас хэтрээгүй эсэхийг шалгах, хэтэрсэн бол зайг тогтоох функц

```
Бичигдэх хэлбэр:
[объект].CheckCollision([цэгийн байрлал],[зай]);
```

Жишээ нь: Машин гэсэн объект байсан гэж бодъё. Камер машинд хагас метрээс илүү ойртож чаддаггүй гэж үзье. (Хэрэв камерийг бөмбөрцөг хэлбэртэй гэж үзвэл энэ хагас метр зай нь камерийн радиус юм.) Энэ орчныг үүсгэхийн тулд камерийг ямар нэг хөдөлгөөн хийх бүрт машинд хагас метрээс илүү ойртсон эсэхийг шалгаад хэрэв тийм бол камерийг машинаас хагас метр зай авсаны дараа ямар байрлалд очих вэ гэдгийг тооцож олон камерын байрлалыг тухайн цэгийн координатаар шинэчлэн тогтооно. Үүний дараа камераар харах буюу дэлгэцэнд харагдалтыг дүрслэх юм. Ингэснээр камер машинд хагас метрээс илүү ойртохгүй байгаа юм шиг харагдах болно. Үүнийг гүйцэтгэхийн тулд:

```
Камер.[Хөдөлгө] ( чиглэл );
Камер.[Байрлалыг тогтоо] ( Машин.CheckCollision(
          Камер.[байрлал], [хагас метр] ) );
Камер.[Дэлгэцэнд дүрслэ] ( );
```

2.10 Гартай ажиллах

KeybInput төрөлтэй объект зарлан түүгээрээ дамжуулж компьютерийн гартай ажиллана.

Зарлах

```
Бичигдэх хэлбэр:
KeybInput [объектын нэр];
```

Үүсгэх

```
Бичигдэх хэлбэр:
[объектын нэр] = new KeybInput();
```

Гарны төлөвийг уншиж хадгалах

Бичигдэх хэлбэр:

```
[объектын нэр].Read();
```

Гарны товч бүрийг дарагдсан, дарагдаагүй эсэх, тухайн агшинд дарагдаж байгаа болон тухайн агшинд дарагдахаа больж байгаа гэсэн 4 төлөвийг тодорхойлон хадгалдаг

Товчны төлөвийг шалгах

Бичигдэх хэлбэр:

```
[объектын нэр].KeyPressed([товч]);
```

Тухайн товч дарагдсан төлөвт байгаа бол бүлийн үнэн утга буцаах функц

```
[объектын нэр].KeyNotPressed([товч]);
```

Тухайн товч дарагдаагүй төлөвт байгаа бол бүлийн үнэн утга буцаах функц

```
[объектын нэр].KeyDown([товч]);
```

Тухайн товч тухайн агшинд дарагдаж байгаа бол бүлийн үнэн утга буцаах функц

```
[объектын нэр].KeyUp([товч]);
```

Тухайн товч тухайн агшинд дарагдаагүй төлөвт шилжиж байгаа бол бүлийн үнэн утга буцаах функц

2.11 Хулганатай ажиллах

MouseInput төрөлтэй объект зарлан түүгээрээ дамжуулж компьютерийн хулганатай ажиллана.

Зарлах

Бичигдэх хэлбэр:

```
MouseInput [объектын нэр];
```

Үүсгэх

Бичигдэх хэлбэр:

```
[объектын нэр] = new MouseInput();
```

Хулганы төлөвийг уншиж хадгалах

Бичигдэх хэлбэр:

```
[объектын нэр].Read();
```

Хулганы хөдөлгөөний шилжилт болон товч бүрийг дарагдсан, дарагдаагүй эсэх, тухайн агшинд дарагдаж байгаа болон тухайн агшинд дарагдахаа больж байгаа гэсэн 4 төлөвийг тодорхойлон хадгалдаг

Хулганы төлөвийг шалгах

Бичигдэх хэлбэр:

```
[объектын нэр].ButtonPressed([товч]);
```

Тухайн товч дарагдсан төлөвт байгаа бол бүлийн үнэн утга буцаах функц

```
[объектын нэр].ButtonNotPressed([товч]);
```

Тухайн товч дарагдаагүй төлөвт байгаа бол бүлийн үнэн утга буцаах функц

```
[объектын нэр].ButtonDown([товч]);
```

Тухайн товч тухайн агшинд дарагдаж байгаа бол бүлийн үнэн утга буцаах функц

```
[объектын нэр].ButtonUp([товч]);
```

Тухайн товч тухайн агшинд дарагдаагүй төлөвт шилжиж байгаа бол бүлийн үнэн утга буцаах функц

```
[объектын нэр].Move();
```

Хулганы хөдөлгөөний тухайн агшин дахь шилжилтийн утгыг Бутархай тоон утгаар буцаах функц

Техник эдийн засгийн үндэслэл

❖ Техник хангамж

График дүрслэлд маш өндөр хэмжээний дэлгэцийн санах ой, процессорын хурд шаарддаг тул хамгийн багаар дараах үзүүлэлттэй компьютер шаардлагатай.

Pentium 4 1.4Ghz+
128MB RAM
Display Adapter AGP GeForce2 32MB+

OpenGL нь сүүлийн үеийн дэлгэцийн картууд дээр л ажиллах боломжтой байдаг. Дүрслэл нь график карт дээр явагддаг учир AGP slot -той Mainboard, дэлгэцийн санах ой хамгийн багадаа л 32MB байх хэрэгтэй.

❖ Программ хангамж

Windows 98+ үйлдлийн систем, дэлгэцийн картын программ хангамжыг зөв суулгасан байх, Microsoft Framework .NET v1.0, DirectX 9.0c суулгасан байх шаардлагатай.

➢ Эдийн засгийн үндэслэл

Ашигласан материал

1. OpenGL Programming Guide (The Red Book) (ISBN 0-321-17348-1)
2. The OpenGL Graphic System: A Specification v1.5
3. OpenGL Reference Manual
4. NeHe Tutorials http://nehe.gamedev.net
5. Game Tutorials http://www.gametutorials.com
6. Apron Tutorials http://www.morrowland.no/apron/
7. Computer Graphics (ISBN 0-07-135781-5)
8. Tricks Of Windows Game Programming Gurus (ISBN 0-672-31361-8)
9. Microsoft DirectX 9.0c SDK Tutorials & Documentation
10. Autodesk .3ds File Format
11. Basic 3D Math http://3dhtml.netzministerium.de/2_basic3dmath.html
12. Vector Math for 3D Computer Graphics
    http://chortle.ccsu.ctstateu.edu/VectorLessons/vectorIndex.html
13. 3D Studio Max v6.0 Tutorials