МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Соёл-Эрдэнэ Мөнх-Очир

Процедурын дагуу видео тоглоомын орчныг үүсгэх

(Procedurally Generating Video Game Enviroment)

Програм Хангамж(D061302)
Бакалаврын судалгааны ажил

Улаанбаатар

2023 оны 3 сар

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
ХЭРЭГЛЭЭНИЙ ШИНЖЛЭХ УХААН, ИНЖЕНЕРЧЛЭЛИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Процедурын дагуу видео тоглоомын орчныг үүсгэх (Procedurally Generating Video Game Enviroment)

Програм Хангамж(D061302)
Бакалаврын судалгааны ажил

Удирдагч: _____ Проф. Х. Оюундолгор

Гүйцэтгэсэн: _____ С. Мөнх-Очир (19B1NUM0574)

Улаанбаатар

2023 оны 3 сар

Зохиогчийн баталгаа

Миний бие Соёл-Эрдэнэ Мөнх-Очир ”Процедурын дагуу видео тоглоомын орчныг үүсгэх” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулж байгаагүй.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дипломын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

ГАРЧИГ

УДИРТГАЛ 1

БҮЛГҮҮД 2

1. ТАНИЛЦУУЛГА 2

1.1 Тоглоомын контент 2

1.2 Тоглоомын явц 3

1.3 PCG-н шийдлийн шинж чанар 3

1.4 PCG гэдэгт дараах зүйлсийг хамруулахгүй 4

2. АЛГОРИТМЫН СУДАЛГАА 6

2.1 Энгийн өрөөний байршуулалт 6

2.2 BSP 6

2.3 Cellular automata 10

3. АШИГЛАХ ТЕХНОЛОГИ 12

3.1 Unity 12

3.2 C# 13

3.3 IntelliJ Rider 13

3.4 PlasticSCM 13

4. ХЭРЭГЖҮҮЛЭХ СИСТЕМ 17

4.1 Хэрэгжүүлэх PCG-н шийдэл 17

4.2 Системийн функциональ шаардлага 17

4.3 Функциональ бус PCG шийдлийн шаардлага 17

4.4 Хэрэгжүүлэлт 18

4.5 Системийн class diagram 18

4.6 Хэрэглэгчийн харилцах хэсэг 21

4.7 Системийн үйл явц 22

5. ҮР ДҮН, ХЭЛЭЛЦҮҮЛЭГ 25

5.1 Төлөөллийн загварыг үүсгэх 25
5.2 Тоглоомын моделийг үүсгэх 26
5.3 CellUnit, CellBundle үүсгэх. 27
5.4 Хэрэглэгчийн тест, сайжруулалт. 29
5.5 Хэлэлцүүлэг 29
6. ДҮГНЭЛТ 32
НОМ ЗҮЙ 33
ХАВСРАЛТ 34
А. КОДЫН ХЭРЭГЖҮҮЛЭЛТ 34
А.1 Системийн хэсэг 34
В. АЖЛЫН ТӨЛӨВЛӨГӨӨ 41

ЗУРГИЙН ЖАГСААЛТ

2.1 Pixel хуваалт 7

2.2 Өрөө тасалгааны бүтцийг BSP-г үүсгэх үйл явц 8

2.3 BSP-н өгөгдлийн бүтцийг ашиглан өрөөнүүдийг коридороор холбох. . . 9

2.4 2 хэмжээст cellular automata-н хэрэгжүүлэлтийн загвар 10

2.5 Тоглоомын орчин, Cellular automata-н хэрэгжүүлэлт 11

3.1 MonoBehaviour амьдралын мөчлөг эхний хэсэг 15

3.2 MonoBehaviour амьдралын мөчлөг хоёр дахь хэсэг 16

4.1 Class Diagramm 20

4.2 Хэрэглэгчийн харилцах хэсэг - CellBundle 21

4.3 Хэрэглэгчийн харилцах хэсэг - ControllerLevelGenerator 21

4.4 Sequence Diagramm эхний хэсэг 23

4.5 Sequence Diagramm хоёр дахь хэсэг 24

5.1 Ур дүнгийн загвар 25

5.2 Ур дүнгийн загвар 26

5.3 Тоглоомын орчин 27

5.4 CellUnit-г тодорхойлж буй жишээ 28

5.5 Хаалга CellUnit-н алдааны засвар 30

B.1 Ажлын төлөвлөгөө 42

Кодын жагсаалт

A.1 BinarySpaceTree хэрэгжүүлэлт 34

УДИРТГАЛ

Видео тоглоом нь өндөр зардлаар бүтдэг, өрсөлдөөн ихтэй салбар юм. Учир нь тоглоомын контент нь тоглоомын түүх, тоглоомын орчин, дүр болон орчны модель, моделийн материал зэрэг олон хүчин зүйлсээс хамаардаг. Тоглоомын контентыг компьютерын тусламжтай үүсгэх нь хүн хүч, цаг хугацааг хэмнэх олон давуу талтай. Тиймээс энэхүү судалгааны ажлаар тоглоомын контентыг процедураар үүсгэх арга техникийг dungeon-crawler төрлийн тоглоомын орчныг үүсгэх жишээн дээр тайлбарлан харуулна.

Түлхүүр үгс: ***Процедурын дагуу контентыг үүсгэх, тоглоомын контент, тоглоомын явц***

Зорилго: PCG-г видео тоглоомын хувьд шинжилгээ судалгаа хийн яагаад орчин үед үүнийг илүү ашиглахгүй байна бэ? мөн тоглоомын явцад нөлөөлөх боломжтой өөр содон арга хэлбэрийн талаар хэлэлцүүлэг эхлүүлэх.

Зорилт:

- Unity дээр C# хэл дээр олон давхар процедурын алгоритм нэгтгэн dungeon-crawler тоглоомын орчныг үүсгэх систем боловсруулах.
  - Алгоритмын судалгаа
  - Харилцах хэсгийг боловсруулах
  - Системийг боловсруулах
  - Туршилт хийх
  - Сайжруулах
- Үр дүнгийн шинжилгээ
  - Үр дүнгийн танилцуулга, шинжилгээ
  - PCG төрлийн контентыг тоглоомын явцад нөлөөлөх байдлаар ашиглах хэлбэрийн хэлэлцүүлэг

1. ТАНИЛЦУУЛГА

Процедурын дагуу контентыг үүсгэх (Procedural Content Generation буюу PCG) гэдэг нь хэрэглэгчийн хязгаарлагдмал болон шууд бус оролцоотойгоор алгоритм ашиглан контентыг үүсгэх явдал юм [3]. Процедурын дагуу контентыг үүсгэх нь богино хугацаанд бага өртгөөр их хэмжээний контентыг үүсгэх боломжийг олгодог. Орчин үед тоглоомын контентыг процедурын дагуу үүсгэсэн олон жишээ бий. Гэхдээ том тоглоомуудын хувьд тоглоомын явцад нөлөөлөх байдал нь энгийн, сулавтар байгаа. Иймээс бид энэхүү өгүүллээр видео тоглоомын хувьд PCG-ийн талаар дэлгэрүүлэн судлан энэ аргыг видео тоглоомын салбарт яагаад өргөнөөр тоглоомын явцад оруулж өгөхгүй байгаа талаар шинжилж, нөөцийг хэмнэхэд өөр ямар боломжууд байгаа талаар дүгнэн хэлэлцэнэ. Өгүүллийн дараагийн хэсгүүдэд dungeon-crawler төрлийн тоглоомын орчныг процедурын дагуу үүсгэх системийн алгоритмын сонголт, гүйцэтгэл, туршилт, эцэст нь туршилтаар олсон үр дүнд тулгуурлан дэвшүүлсэн асуултад хариулж, хэлэлцүүлэг, дүгнэлт хийнэ. Энэхүү судалгааны ажилд dungeon-crawler төрлийн тоглоомын орчныг хэрхэн үүсгэхийг авч үзэх бөгөөд үүнд олон өрөө, тасалгаатай орчин багтана.

1.1 Тоглоомын контент

Тоглоомын контентод тоглоомын түүх, тоглоомын орчин, модель, моделийн материал зэрэг орж болох ба тоглоомыг хийхэд ашигласан engine, цаад систем зэрэг нь орохгүй. Өөрөөр хэлбэл тоглоомын контент нь тоглоомыг бүрэлдүүлж буй системүүдийн үр дүн, тоглогч тоглоомтой харилцах хэлбэр юм.

Тоглоомын контентыг компьютерын тусламжтайгаар үүсгэвэл артист, хөгжүүлэгч нарын цагийг ихээр хэмнэн нөөцийг үр ашигтай ашиглах боломжтой. Видео тоглоомыг үйлдвэрлэхэд үнэ өртөг жил ирэх тусам өссөөр байгаа ба уг салбар нь асар том юм. Хүн хүч болон цаг хугацаа нь өртөг ихтэй ба энэхүү өрсөлдөөн ихтэй салбарт нөөцийг үр ашигтай ашиглавал тухайн бүтээгдэхүүн нь амжилттай байх магадлал нь нэмэгдэнэ харин PCG-г ашигласнаар нөөцийг илүү үр дүнтэйгээр ашиглан илүү сайн, бүтээлч бүтээгдэхүүнийг үйлдвэрлэх боломжтой.

PCG-г орчин үед том тоглоомууд ихээр ашиглаж байгаа гэхдээ ашиглах хэлбэр нь энгийн, тоглоомын явцад нөлөөлөх хэлбэр нь сул байдаг. Үүнд орчин үеийн AAA тоглоомуудын модны моделийг процедурын аргаар үүсгэдэг гол дундын систем болох SpeedTree, процедурын дагуу үүссэн моделийн материалын жишээ зэрэг орж болно. Модны модель болон материал зэрэг нь тоглоомын контент мөн гэхдээ тоглоомын явцад нөлөөлөх нь бага.

1.2 Тоглоомын явц

Тоглоомын явц нь тоглогч тоглоомтой харилцах гол хэлбэр ба тухайн тоглоомоос шалтгаалаад энэ нь янз бүр байж болно. Тоглогч нь тоглоомтой олон хэлбэрээр харилцаж болно харин тэр болгон нь тоглоомын явцад орохгүй. Мэдээж бүх тоглоомд гол цэс эсвэл хэрэглэгчийн харилцах хэсэг байдаг харин тухайн тоглоом нь хэрэглэгчийн харилцах хэсгээрээ тоглогчоо татаж эсвэл энэ нь гол тоглогдох хэлбэр нь биш л бол үүнийг тоглоомын явцад нөлөөлж байна гэж үзэхэд хэцүү. Тоглоомын явцад харин Minecraft тоглоомын эрх чөлөөтэйгөөр блокуудыг удирдах орж болно. Тоглоом нь тоглоомын явцаар тоглогчийг уусган, өөрийг нь илэрхийлэх хэлбэрийг тодорхойлдог. Тоглоомд тоглоомын явц чухал нөлөөтэй.

Тоглоомын явцад нөлөөлөх контентуудыг харин процедурын дагуу үүсгэвэл тоглоомын явцыг процедурын дагуу үүсгэх боломжтой. Ингэснээр тоглоомын дахин тоглогдох чадвар сайжирч тоглолт болгон нь шинэ содон өөр байх боломжтой ба энэ нь бүтээгдэхүүний үнэ цэнд сайнаар нөлөөлнө.

1.3 PCG-н шийдлийн шинж чанар

PCG нь нэг талдаа компьютер болон алгоримтийн тусламжтайгаар хөгжүүлэгчийн оролцоог багасган контентыг үүсгэх үйл явц. Үүнд тухайн контентыг үүсгэх үйл явцын шийдэл хамаарагдана.

1.3.1 PCG болон үүний шийдлийн хүсэмжид шинж чанарууд[2]:

- Хурд

Зарим тоглоомын контентыг ажиллах хугацааны явцад эвсэл тоглоомын хөгжүүлэлтийн явцад ашиглагдахаас шалтгаалаад үүсгэх хугацаа нь янз бүр байж болно. Тухайлбал тоглоомын орчин нь тоглогчийн хариу үйлдлээс хамаарч үүсгэгддэг бол шийдлийн хурд нь 2 милл секундийн дотор байна.

- Найдвартай байдал

  Мөн тухайн контентын шаардлагаас хамааraад энэхүү шинж чанарын биелэгдэх байдал өөр байна. Тоглоомын орчны ердийн өвс эвсэл цэцгийг үүсгэхэд үүний зөв эсэх нь чухал биш харин тоглоомын орчны өрөөнүүдийн ерөнхий бүтцийг гаргаж байвал аливаа өрөө нь орох гарах багадаа хоёр хаалга байх зэрэг шаардлага байж болно.

- Удирдах чадвар

  Энд тухайн шийдлийн үр дүн буюу контентын шинж чанарыг хэрэглэгч өөрчлөх эсвэл удирдах орно. Жишээлбэл өвсний урт өргөн, өнгө зэргийг хэрэглэгчид төвөг удахгүй өөрчлөх орж болно.

- Илэрхийлэлт болон олон талт байдал

  Аливаа нэг сэдэвтэй олон төрлийн контентыг үүсгэх шаардлага тоглоомд тулгарч байдаг. Жишээлбэл тоглоомын орчны өрөөг үүсгэхэд олон талт байдлыг хэрэгжүүлж өгснөөр тухайн өрөөний дотор засал бусад өрөөнөөс өөр байх зэрэг орж болно.

- Бүтээлч байдал болон итгэж болох байдал

  Процедурт аргаар үүсгэгдсэн контентод тоглогч итгэж болох байх шаардлагатай. Жишээлбэл тоглоомын орчны өрөө хэт том биш эсвэл том бол дотроо тулгуур баганатай байх зэрэг орж болно.

1.4 PCG гэдэгт дараах зүйлсийг хамруулахгүй

- Санамсаргүй эсэх

  - PCG-н олон талт байдлаас шалтгаалж хүмүүс PCG-г санамсаргүй гэж бодох тохиолдол байдаг. Гэхдээ энэ нь олон алгоритм болон дүрмийн боловсруулалт ба санамсаргүй биш юм.

- Автомат эсэх
  - PCG-д хэрэглэгчийн хариу үйлдлээс шалтгаалж боловсруулалт хийдэг шийдлүүд байдаг ба энэ нь автомат биш юм.

2. АЛГОРИТМЫН СУДАЛГАА

2.1 Энгийн өрөөний байршуулалт

Олон өрөө тасалгаатай орчныг тоглоомд үүсгэдэг энгийн арга. Процедурт аргаар тоглоомын орчныг үүсгэж сурах гэж байгаа бол энэ аргаас эхэлбэл тохиромжтой.

2.1.1 Энгийн өрөө байршуулалтын хэрэгжүүлэлт [4]

1. Орчныг ханаар дүүргэнэ
2. Санамсаргүйгээр өрөөний байршлыг сонгоно.
   (a) Хэрвээ энд аль хэдийн өрөө байхгүй бол өрөөг нэмнэ.
3. Шаардлагад хүрэх хүртэл алхам 2-г давтана.
4. Нэмсэн өрөөнүүдээ коридороор холбоно.

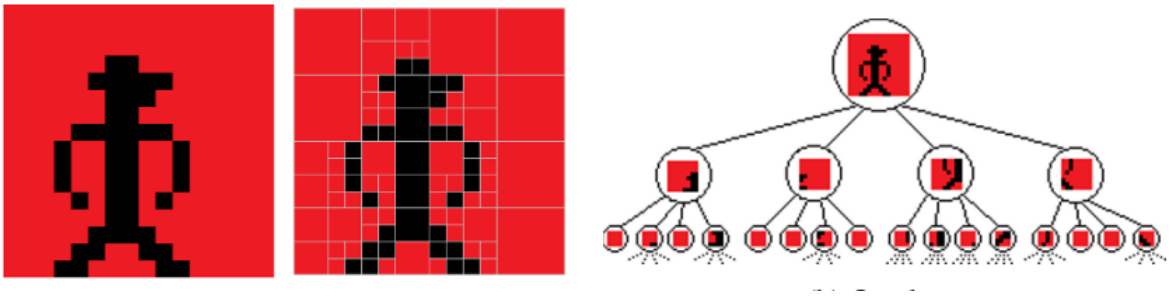
2.2 BSP

Энгийн өрөөний байршуулалттай төстэй ба бүтэн нэр нь Binary Space Partitioning. Энэ аргад орон зайг partitioning хийх буюу орон зайны хуваалт хийдэг. Орон зайг хуваах алгоритмууд ихэвчлэн шаталсан байдлаар ажилладаг. Орон зайн хуваалт нь нүд бүрийг ижил алгоритмаар рекурсив байдлаар хуваах ба энэ нь өгөгдлийг орон зайг space-partitioning tree буюу орон зайн хуваалтын мод бүтцэд зохион байгуулах боломжийг олгоно.

Space-Partitioning Tree

Уг бүтцээр орон зайн доторх аль ч цэгийн геометр хайлтыг хурдан хийдэг. Энэ нь компьютер графикт Space-Partitioning Tree бүтцийг онцгой ач холбогдолтой болгох ба raycasting, frustum culling мөн collision detection-г үр ашигтай хийх боломжийг олгоно.

Зураг 2.1: Pixel хуваалт



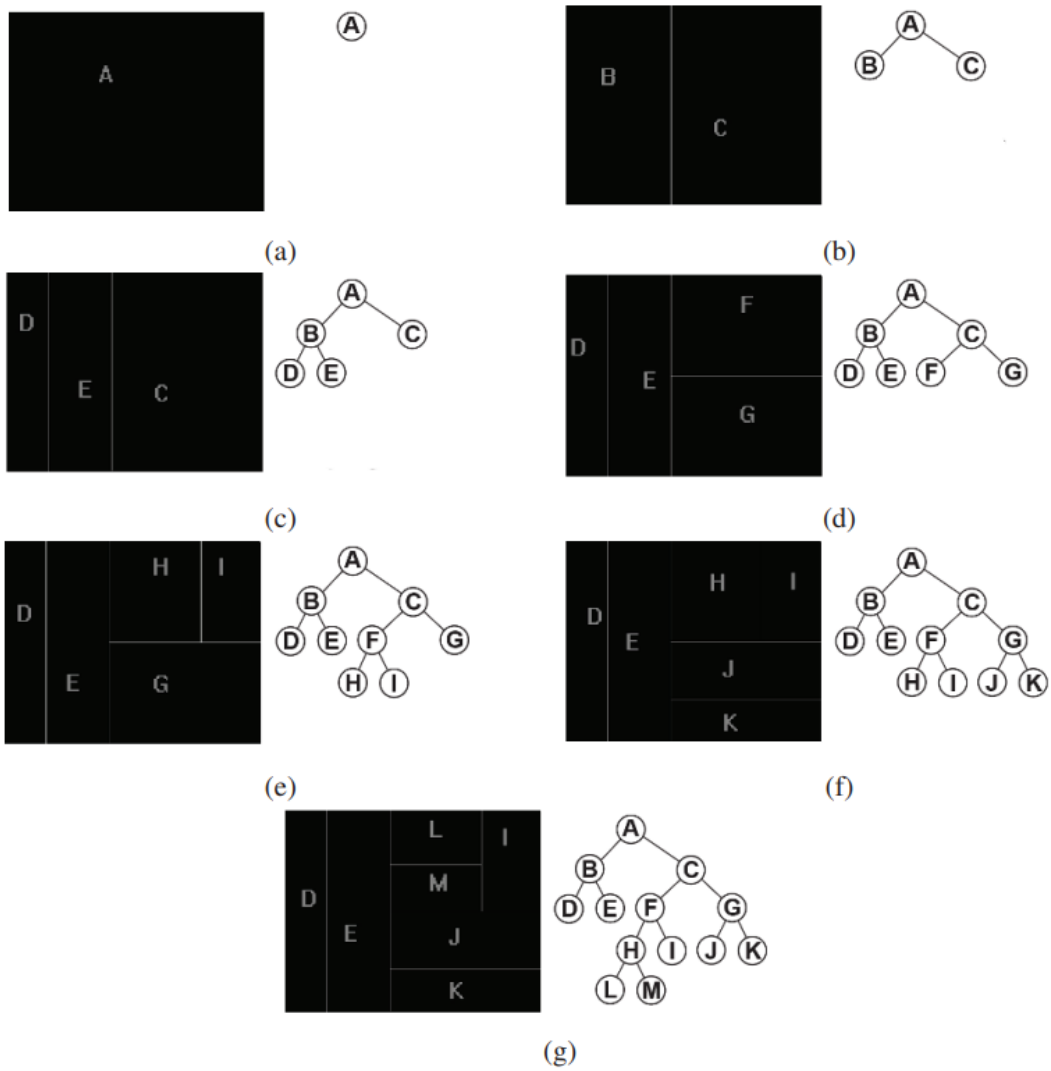
Хамгийн түгээмэл орон зайн хуваалтын арга нь Binary Space Tree. Уг аргад орон зайн нүдийг рекурсив байдлаар хоёр хуваадаг. Ингэснээр орон зай нь Binary Tree буюу хоёртын модон хэлбэрээр төлөөлөгдөх боломжийг олгодог. Хоёртын мод нь цаашлаад үр ашигтай өгөгдлийн хайлт, хадгалалт зэргийг хийх боломжийг олгоно. Space-Partitioning-д хоёртын модноос гадна дөрөвтийн мод эсвэл наймтын мод зэрэг өөр хэлбэрээр хэрэгжүүлж болдог.

Space-Partitioning-г зургийн pixel-н мэдээллийг үр ашигтай хадгалахад ашиглаж болно Зураг 2.1.

Зураг 2.1-д: хоёртын зургийг орон зайн дөрөвтийн хуваалт хийсэн. Нэг өнгийн томоохон хэсгүүд болох баруун талын хэсэг нь цаашид хуваалт хийгдээгүй байна. Зураг нь 16x16 харьцаатай ба дөрөвтийн модны гүн нь дөрөв(дөрөвтийн мод 2^n хэмжээтэй мэдээллийг илтгэж чадна. Энд n-н нь модны гүн) Space-Partitioning аргын давхардсан орон зайн хэсэг үүсгэдэггүй, орон зайг салгадаг зарчим нь олон өрөө тасалгааны бүтцийг гаргахад үр дүнтэй арга [2].

Уг арга нь тоглоомын орчныг үүсгэх макро арга ба дээрээс доошоо ажиллах зарчимтай. Өөрөөр хэлбэл уг шийдэл нь орчны хэмжээ хязгаарын мэдээлэл дээр тулгуурлаж хуваалтаа хийдэг.

2.2.1 Өрөө тасалгааг BSP-р үүсгэх үйл явц



Зураг 2.2: Өрөө тасалгааны бүтцийг BSP-г үүсгэх үйл явц

1. Орчны орон зайг бүхлээр нь авна - BSP-н үндэс хэсэг.

2. Орон зайг босоо эсвэл хэвтээгээр хуваана.

3. Хуваагдсан хэсгүүдийн нэгийг нь сонгоно

4. Хэрвээ уг хэсэг нь хамгийн жижиг хэмжээнээс их бол

5. уг хэсгээр 2 дахь алхам руу явна.

6. Нөгөө хэсгийг сонгоод 4 дэх алхам руу явна.

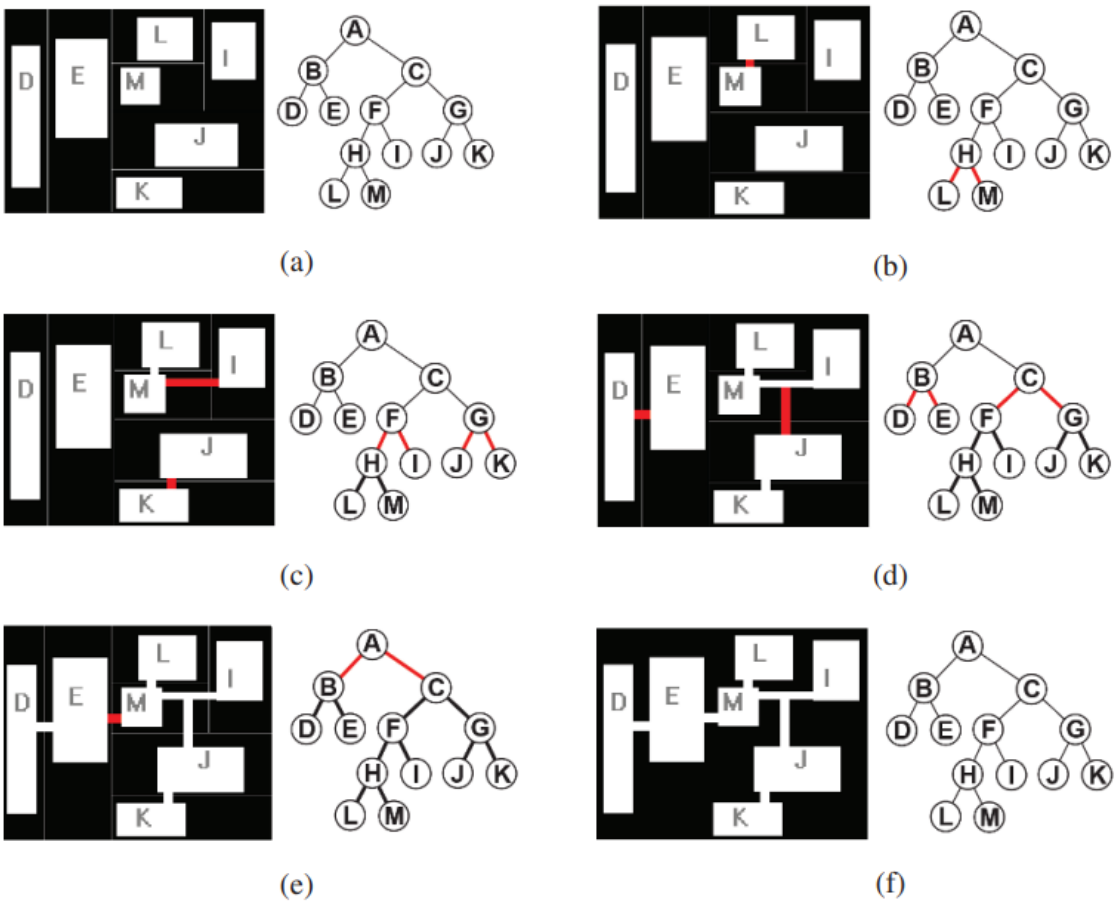
7. Бүх хуваалтын хэсэг болгонд:

8. Хэсгийн хил дотор санамсаргүйгээр өрөө үүсгэнэ.

9. Хамгийн доод талын давхаргаас эхлээд адилхан эцэгтэй хэсгүүдийг хооронд нь коридороор холбоно.

10. Бүх хэсэг холбогдох хүртэл алхам 9-г давтана.

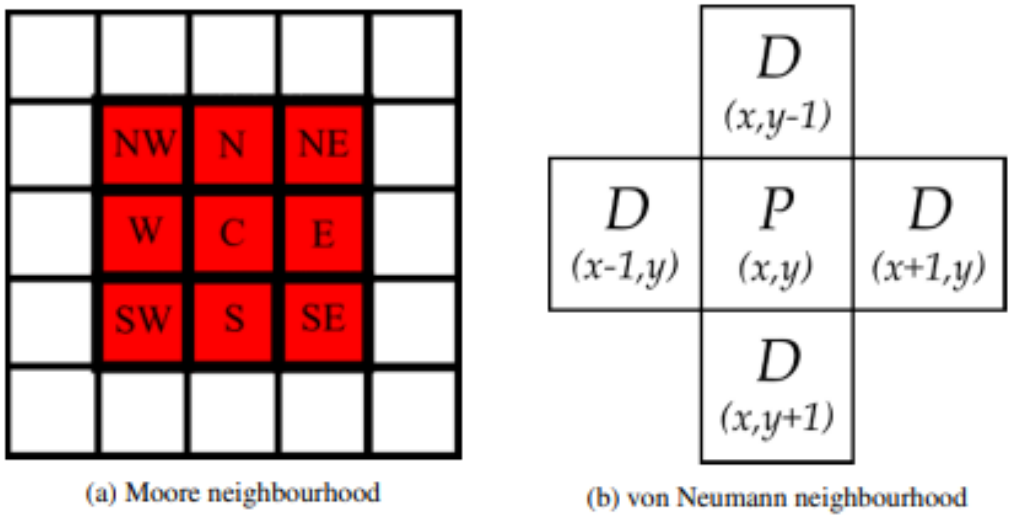
Зураг 2.3: BSP-н өгөгдлийн бүтцийг ашиглан өрөөнүүдийг коридороор холбох.



Зураг 2.3-н зарчмаар өрөөнүүдийг холбосноор хаалганы оноолт нь зөв, аль ч өрөөнөөс өөр аль ч өрөө рүү явах боломжийг олгоно[2].

2.3 Cellular automata

Зураг 2.4: 2 хэмжээст cellular automata-н хэрэгжүүлэлтийн загвар



(a) Moore neighbourhood (b) von Neumann neighbourhood

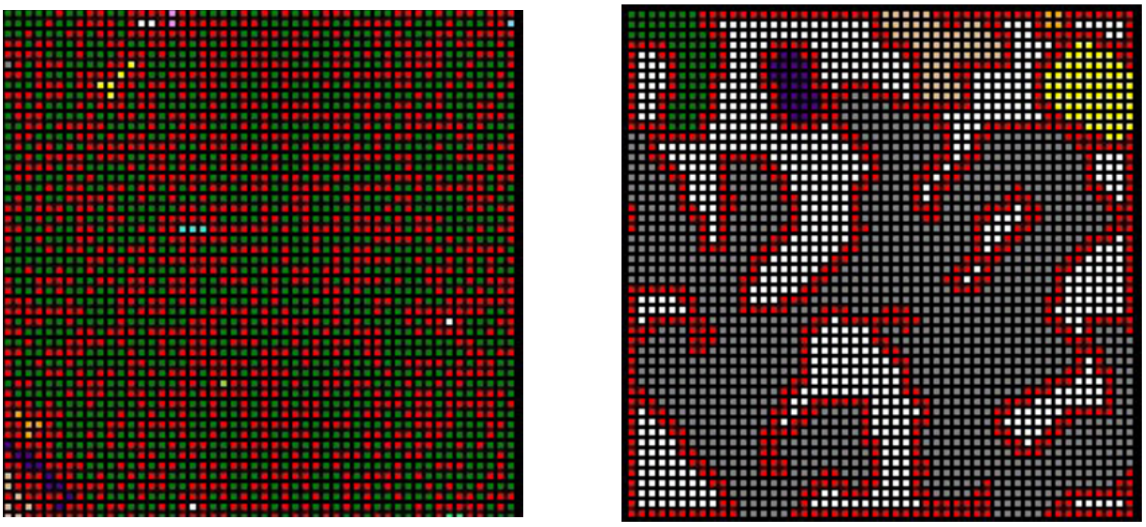
Cellular Automata-н салангид тооцооллын загвар ба компьютерын ухаан, биологи, физик олон шинжлэх ухааны салбарт өргөн хэлбэрээр судалсан байдаг. Гол ухагдахуун болон алгоритм нь цөөхөн алхамтай.

Жишээ нь: Cell буюу нэгж нь хоёр төлөвтэй ба үүнд хана эсвэл хоосон зай байна гэж үзье. n алхмын турш аливаа нэгж нь хөрш нэгжээсээ шалтгаалж өөрийнхөө төлөвийг өөрчилнө. Түгээмэл хоёр хэмжээст cell automata-н хэрэгжүүлэлтийн загварыг Зураг 2.4-г үзүүлэв. Зураг 2.4b-д[2] P-нь одоогийн нэгж ба D нь хөрш нэгж, нэгжийн төлөвийг тодорхойлохдоо хана үүсэх нөхцөлийг 5-с($T = 5$) доошгүй хөрш нэгж нь хана байвал P нэгж өөрөө хана болно гэж тодорхойлж болно. Гэхдээ уг давталтыг хийхийн өмнө орчны нэгжүүдэд санамсаргүй байдлаар хана эсвэл хоосон зайн төлөвийг оноож өгөх хэрэгтэй ба үүнд аливаа нэгж нь хана

болох боломж $r = 0.5$ гэж үзэж болно.

Johnson et al. [1] дээр $n = 4, T = 5, r = 0.5, M = 1$(Улаан ханы зузаан) хувьсагчуудаар Cellular Automata ашиглан тоглоомыг орчныг үүсгэсэн Зураг 2.5. Зураг 2.5a дээр нэгжүүдэд анхны санамсаргүй байдлаар төлвүүдийг оноож өгсөн, Зураг 2.5b дээр Cellular Automata-н үйл явцыг $n = 4$ удаа давтсаны дараах үр дүн.

Зураг 2.5: Тоглоомын орчин, Cellular automata-н хэрэгжүүлэлт



(a) Random map (b) CA map

Уг аргаар органик мэт газар доорх тоглоомын орчныг үүсгэх боломжтой. Гэхдээ энэ аргын нэг дутагталтай тал нь хувьсагчуудын утгыг өөрчлөхөд үр дүнг урьчилан мэдэхэд хэцүү.

3. АШИГЛАХ ТЕХНОЛОГИ

3.1 Unity

Unity нь видео тоглоом, симуляци болон бусад интерактив контентыг үүсгэх, хөгжүүлэхэд ашигладаг cross-platform тоглоомын engine юм. Уг тоглоомын engine нь физик, график, скрипт, нөөцийн менежмент гэх мэт тоглоом хөгжүүлэх олон төрлийн хэрэгсэл, функцээр хангадаг.

3.1.1 Scene

Unity-д Scene нь тоглоомын программын элементүүдийн агуулах ба үүнд GameObjects, гэрэлтүүлэг, камерын тохиргоо болон бусад нөөц орно . Scene-г тоглоомын бие даасан хэсэг эсвэл үе гэж үзэж болох бөгөөд олон Scene-г нэг доор ажиллуулж болдог. Scene-үүдийг динамикаар ачаалж, буулгах боломжтой.

3.1.2 GameObject

Unity-д GameObject нь үзэгдэл дэх тоглоомын объектыг төлөөлөх үндсэн объект юм. GameObject-нь үүнд гол үүрэг, зан чанарыг тодорхойлдог Component-уудын агуулах юм.

3.1.3 Component

Component нь Unity дахь GameObject-д тодорхой функцүүдийг өгдөг модульчлагдсан кодын хэсэг юм. Component-д скрипт, физикийн коллайдерууд, моделийн материалууд орно. GameObject-ийн үйлдлийг өөрчлөхийн тулд бүрэлдэхүүн хэсгүүдийг нэмж, устгаж, өөрчилж болно.

3.1.4 Prefab

Prefab нь GameObject болон түүнтэй холбоотой component-уудыг агуулсан дахин ашиглах боломжтой нөөц бөгөөд Scene-д олон удаа үүсгэж болно. Prefab-ууд нь хөгжүүлэгчдэд объектыг

нэг удаа үүсгэж, тоглоом эсвэл программынхаа туршид дахин ашиглах боломжийг олгодог.

3.1.5 ScriptableObject

Unity дээр өөр дахин ашиглах боломжтой нөөц ба энэ нь Prefab-с Scene-с гадуур ашиглагдаж чаддагаараа өөр.

3.1.6 MonoBehaviour болон үүний амьдралын мөчлөг

MonoBehaviour нь скриптүүдийг Unity engine-тэй харилцах боломжийг олгодог component юм. Component-н функциональ зан чанарыг тодорхойлоход ашиглаж болох Start(), Update() болон OnTriggerEnter() зэрэг функцүүдээр хангадаг. Амьдралын мөчлөгийг 3.1 үзүүлэв.

3.1.7 ProBuilder

Unity дээр моделийг үүсгэх, өөрчлөх мөн тоглоомыг үеийг засварлах хэрэгсэл.

3.2 C#

Кодын хэл ба Unity дээр Component-г тодорхойлоход ашиглана.

3.3 IntelliJ Rider

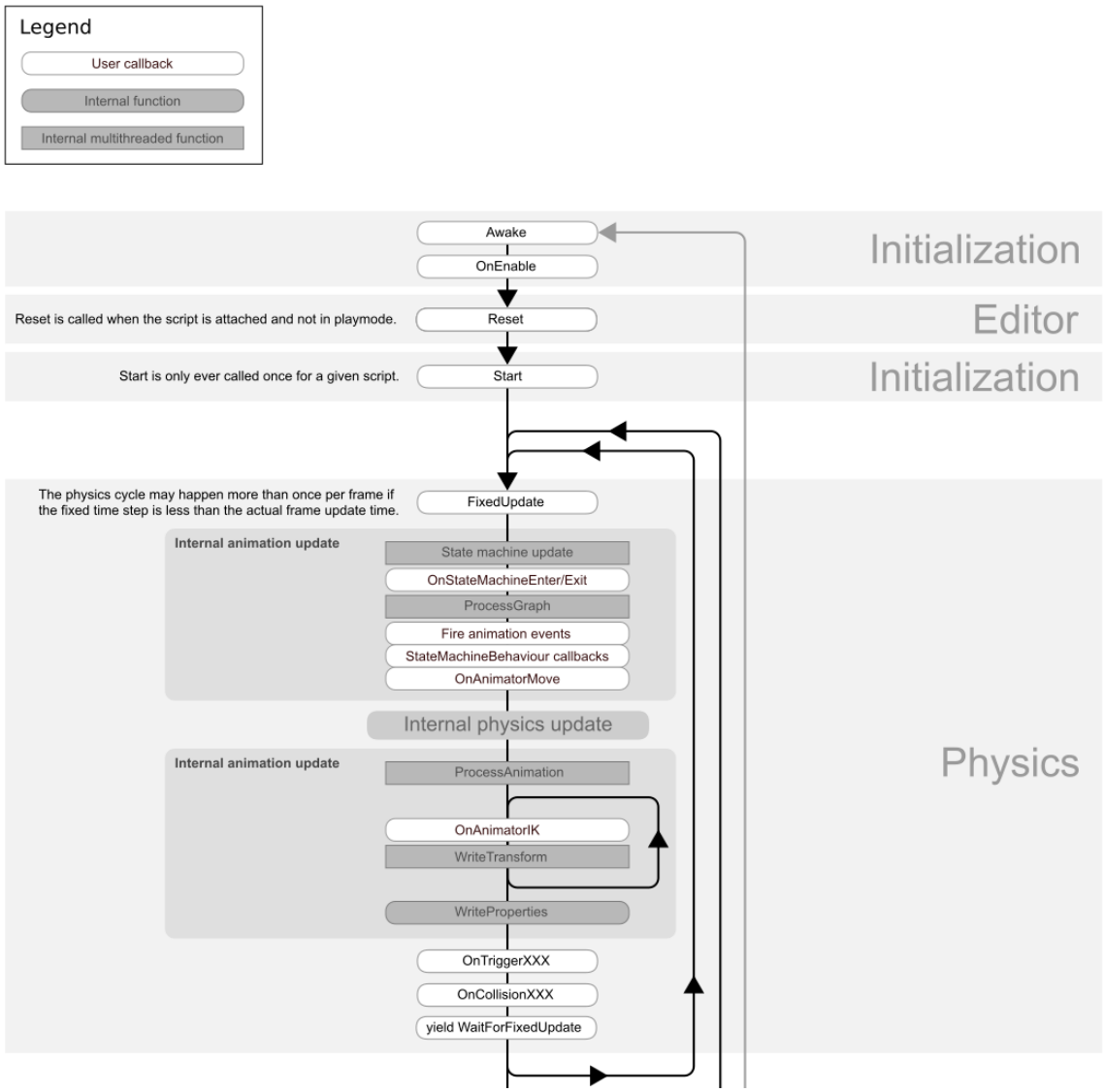
Unity-тэй сайн холболттой. Код highlighting, refactoring, debugging, completion зэрэг функцуудтай IDE.

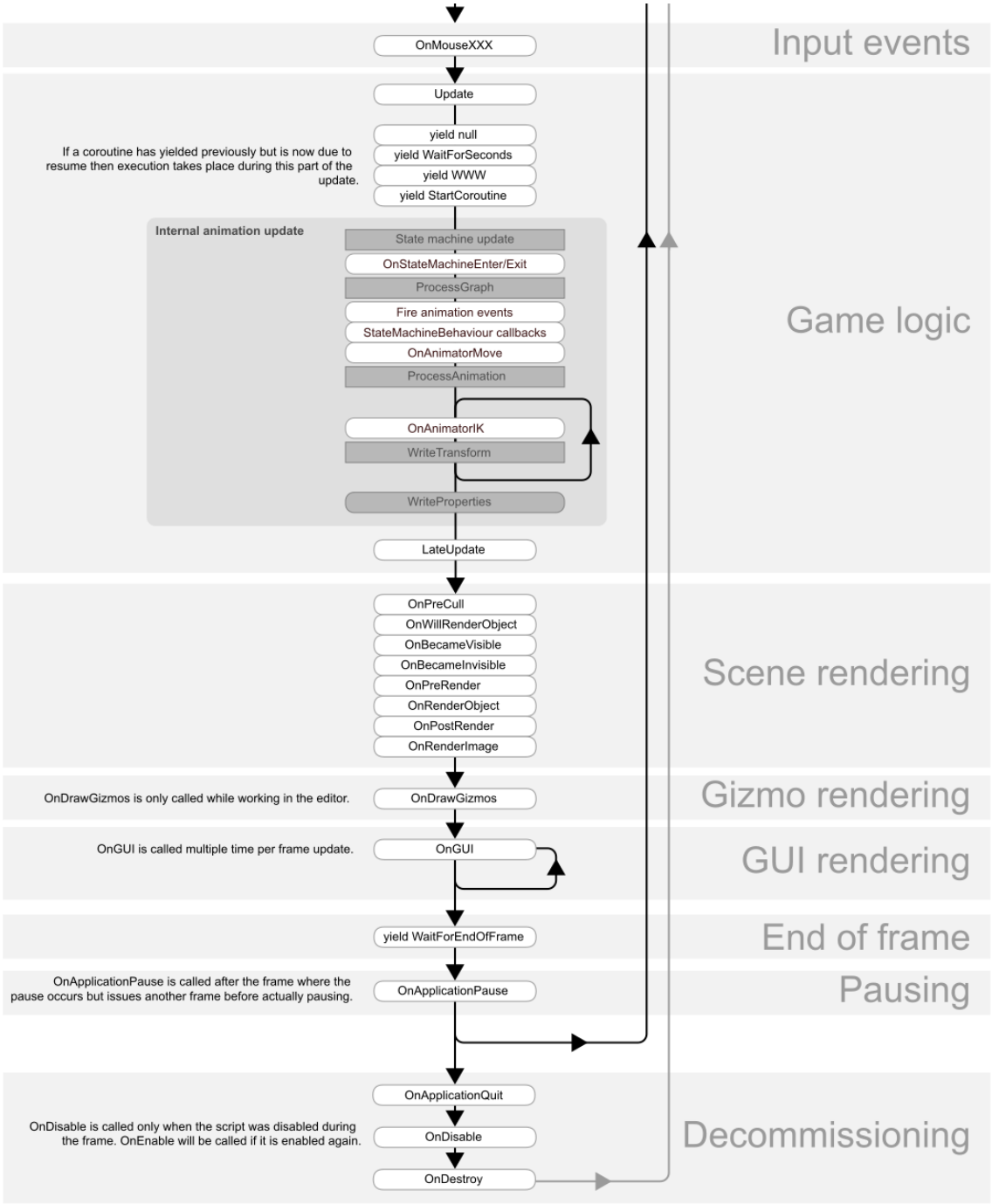
3.4 PlasticSCM

Тоглоомын төслийн хөгжүүлэлтэд нийтлэг ашиглагддаг Version Control хэрэгсэл. Нийтлэг branching, merging, code reviews зэрэг version control хэрэгслүүдийн функционалийг агуулдаг. Харин PlasticSCM нь бусад version control-тэй харьцуулахад том файлуудтай болон тоглоомын

хөгжүүлэлтийн нөөцүүдтэй харьцахад илүү амар.

Зураг 3.1: MonoBehaviour амьдралын мөчлөг эхний хэсэг





Зураг 3.2: MonoBehaviour амьдралын мөчлөг хоёр дахь хэсэг

4. ХЭРЭГЖҮҮЛЭХ СИСТЕМ

4.1 Хэрэгжүүлэх PCG-н шийдэл

Энэхүү бичвэрийн хүрээнд Dungeon-Crawler тоглоомын орчин үүсгэнэ. Системийн нэр ”ControllerLevelGenerator” ба энэ нь Unity-н component.

4.1.1 Dungeon-Crawler төрлийн тоглоом

Энэхүү төрлийн тоглоом нь газар доор эсвэл олон өрөө тасалгаатай орчинд тоглогчийг олон янзаар(ихэвчлэн мангасуудын эсрэг тулалдах, аливаа оньсого тайлах, дүрээ сайжруулах зэрэг) сорьдог тоглоомын төрөл. Тоглоомын орчин нь хязгаарлагдмал, тоглогчийн хийх үйлдэлд нөлөөлдөг тул тоглоомын орчин нь тоглоомын явцад нөлөөлөх байдал нь өндөр. Ийм төрлийн тоглоомд PCG-г ашигласан олон жишээ бий(Rogue 1985, Spelunky 2008, The Binding of the Isaac 2014).

4.2 Системийн функциональ шаардлага

- Хэрэглэгч - ControllerLevelGenerator болон CellBundle-н тохиргоог хийх, шинээр Cell-Bundle үүсгэх, шинээр CellUnit үүсгэх мөн эдгээрийг өөрчлөх.

4.3 Функциональ бус PCG шийдлийн шаардлага

- Хурд

  Орчныг нэг удаа тоглолт эхлэхэд үүсгэнэ. 2 секундийн дотор үүсгэнэ.

- Найдвартай байдал

  Өрөө хоорондын хаалганы байршил нь зөв нэг өрөөнөөс аль ч бусад өрөө рүү дамжин явах боломжтой.

- Удирдах чадвар

  Өрөөний үзэмж, орчны элементүүдийн тодорхойлох болон элементийн сонгогдох боломжийг удирдах боломжтой.

- Илэрхийлэлт болон олон талт байдал

  Өөр хэмжээтэй өрөөнүүд, өрөөний элементүүд нь олон сонголттой байх.

- Бүтээлч байдал болон итгэж болох байдал

  Өрөө нь хэт том эсвэл хэт жижиг биш. Хаалга болон шатны байршил нь тоглогч итгэж болох байх.

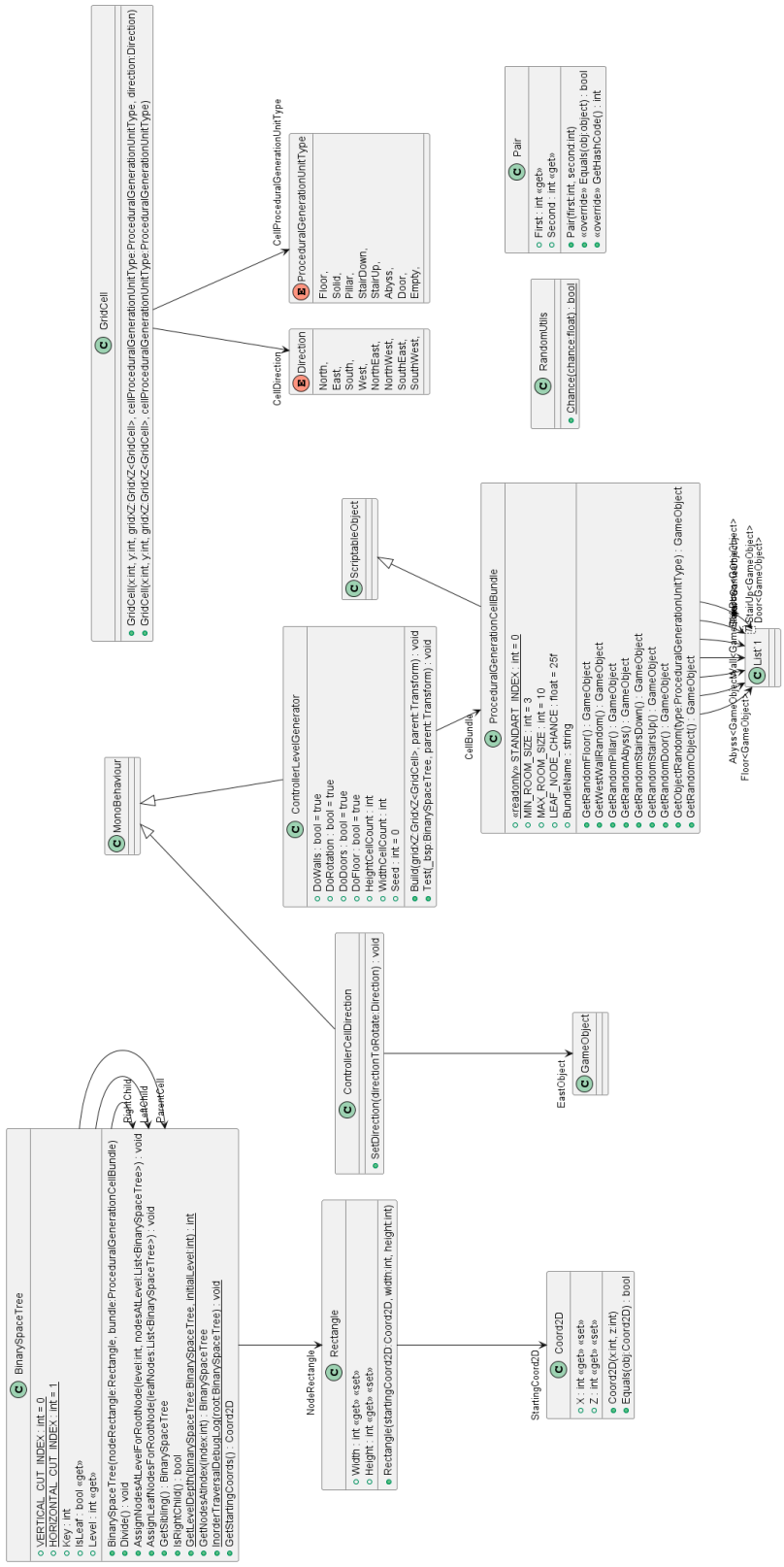
4.4 Хэрэгжүүлэлт

4.5 Системийн class diagram

- Rectangle - Хоёр хэмжээст тэгш өнцөгтийг илэрхийлнэ.
- GridXZ - Гурав хэмжээсд ашиглах хоёр хэмжээст торон систем.
- GridCell - GridXZ-н нэгж
- CellBundle - Дотроо нэг сэдэвтэй CellUnit-үүдийн багц.
- CellUnit - Тухайн төрлөө илтгэх модель болон component-тэй prefab.
  - Floor, Solid, Pillar, StairDown, StairUp, Abyss, Door, Empty гэсэн төрлүүдтэй.
  - 5м^2 харьцах хэмжээтэй орчны нэг нэгж.
- ControllerLevelGenerator - Тоглоомын орчныг үүсгэх хэрэглэгчтэй харилцах component, харилцах систем.
  - Санамсаргүй тооцооллын Seed авдаг ба уг утгаар системийн орчны үр дүнг дахин боловсруулж болно.

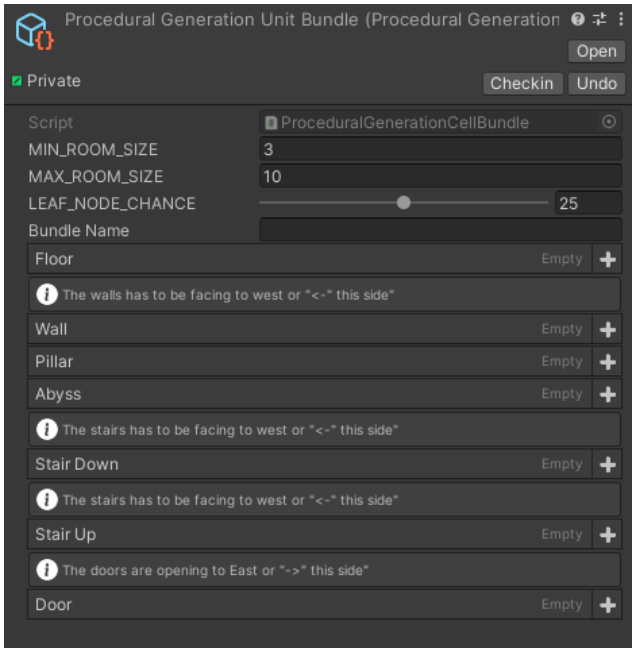
– Scene эхлэх үед тоглоомын орчныг үүсэх үйл явцыг эхлүүлэн системийг гол удирдах, хянах үүргийг гүйцэтгэдэг хэсэг.

Зураг 4.1: Class Diagramm



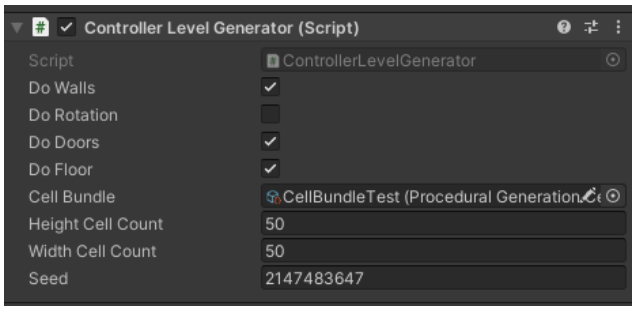
4.6 Хэрэглэгчийн харилцах хэсэг

4.6.1 CellBundle



Зураг 4.2: Хэрэглэгчийн харилцах хэсэг - CellBundle

4.6.2 ControllerLevelGenerator



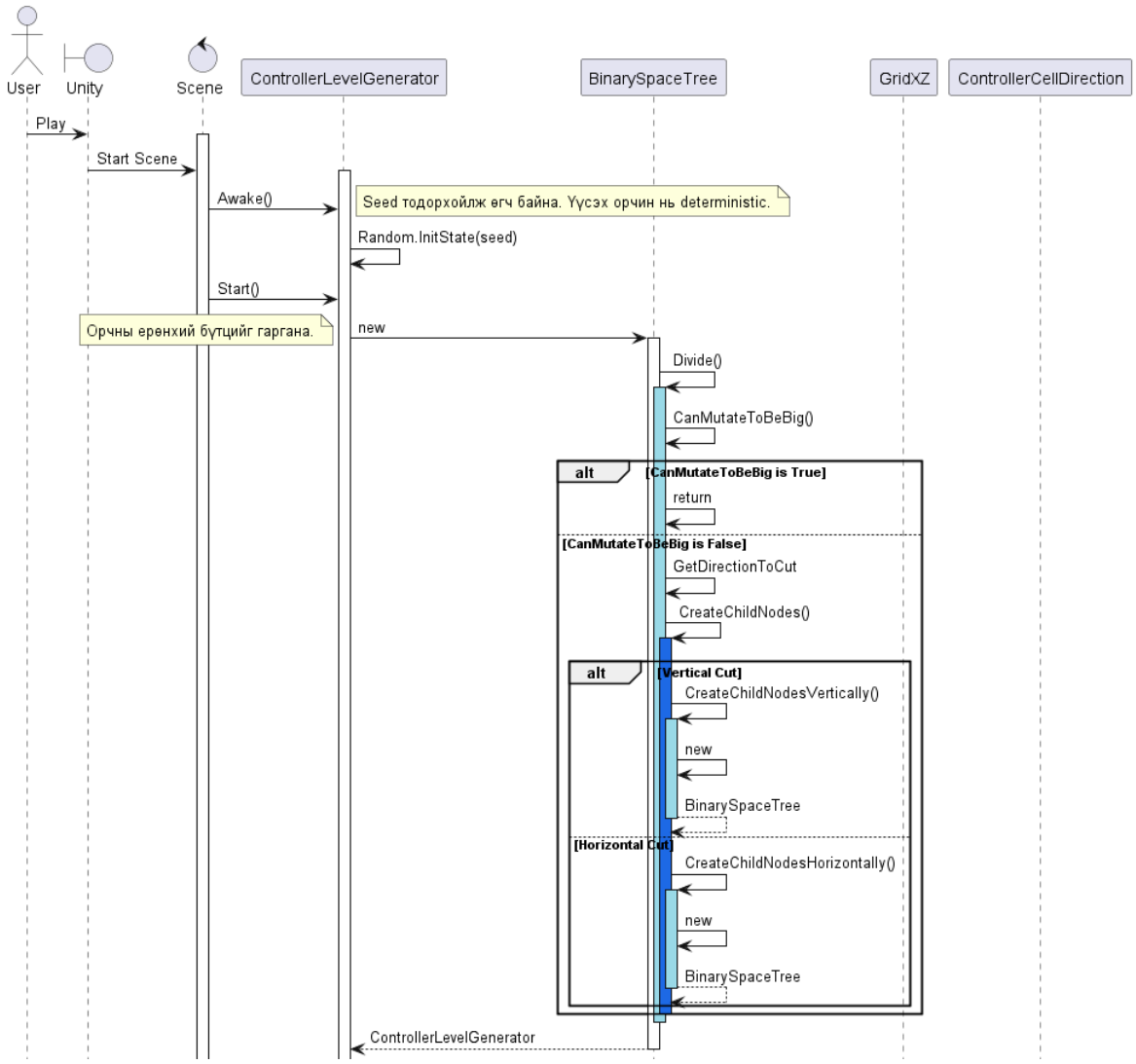
Зураг 4.3: Хэрэглэгчийн харилцах хэсэг - ControllerLevelGenerator

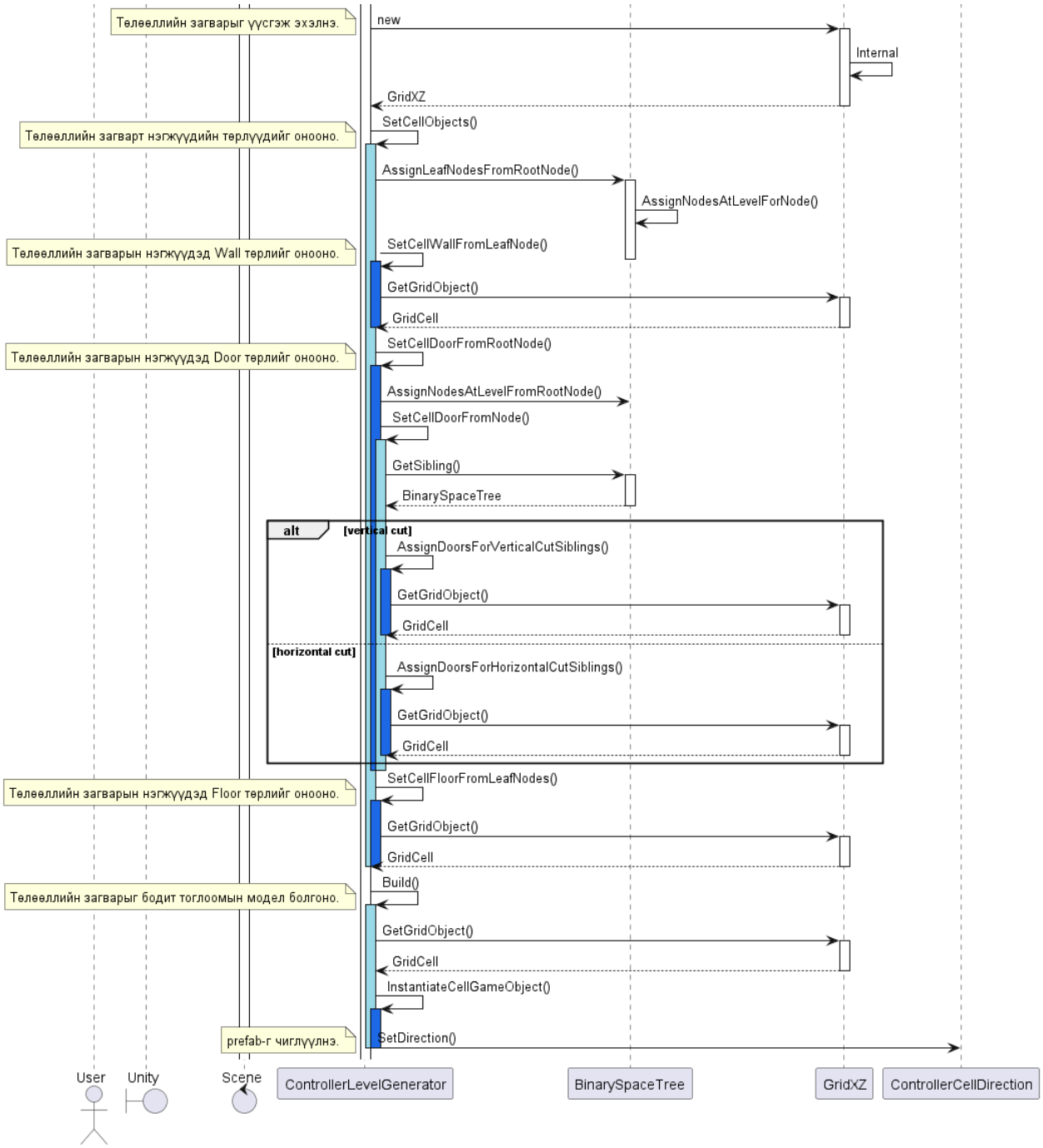
4.7 Системийн үйл явц

Dungeon-crawler төрлийн тоглоомны орчныг процедурт шийдлээр үүсгэхэд ерөнхийдөө дараах гурван зүйлсийг хийдэг ба дараах байдлаар гүйцэтгэв.

- Төлөөллийн загвар. Орчны өрөө тасалгааны ерөнхий бүтцийн энгийн загвар.
  - Үүнийг хоёр хэмжээст торон системээр хэрэгжүүлнэ.
  - Торон системийн нэгж нь GridCell
- Төлөөллийн загварыг үүсгэх үйл явц буюу арга.
  - Орчны ерөнхий бүтцийг BSP ашиглаж гаргана. Үүнд өрөөнүүдийн хэмжээ болон байршил нь торон системдээр тодорхойлогдоно.
  - Орчны ерөнхий бүтэц дээр нэмэлт өөрчлөлт хийнэ. Үүнд Floor, Door, Wall CellUnit үүдийг торон системд олгоно.
- Төлөөллийн загварыг бодит тоглоомын модел болгох үйл явц буюу арга.
  - ControllerLevelGenerator нь GridXZ-н GridCell болгон дээр төрөл болон байршлын мэдээлэл дээр тулгуурлан CellBundle-с prefab-г Scene дээр үүсгэнэ.

Зураг 4.4: Sequence Diagramm эхний хэсэг

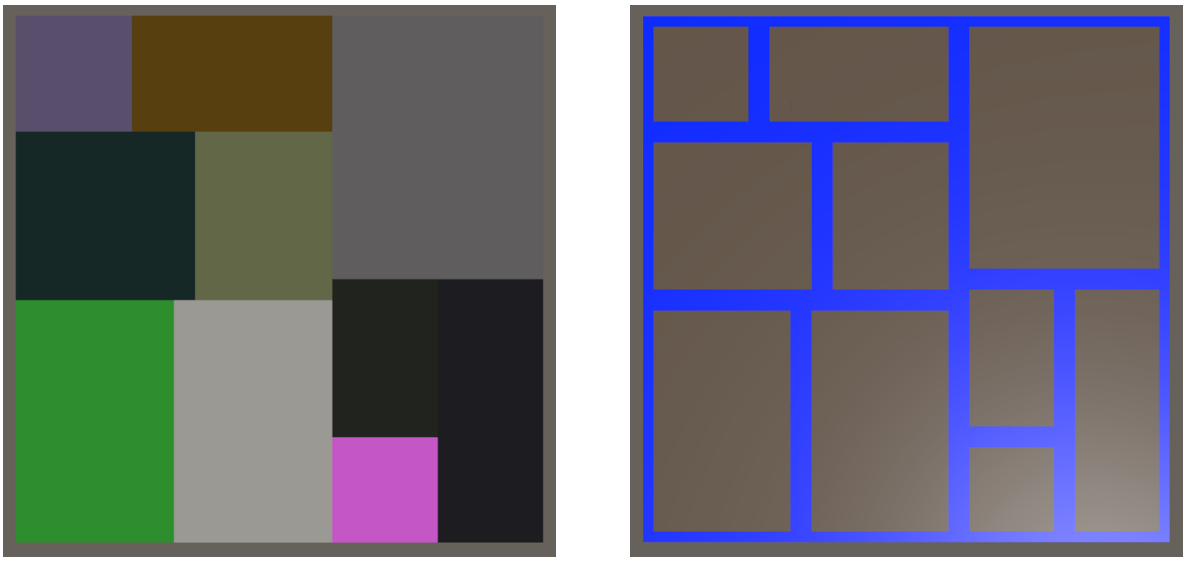




Зураг 4.5: Sequence Diagramm хоёр дахь хэсэг

5. ҮР ДҮН, ХЭЛЭЛЦҮҮЛЭГ

5.1 Төлөөллийн загварыг үүсгэх



(a) BSP (b) Хануудыг тодорхойлсон GridXZ

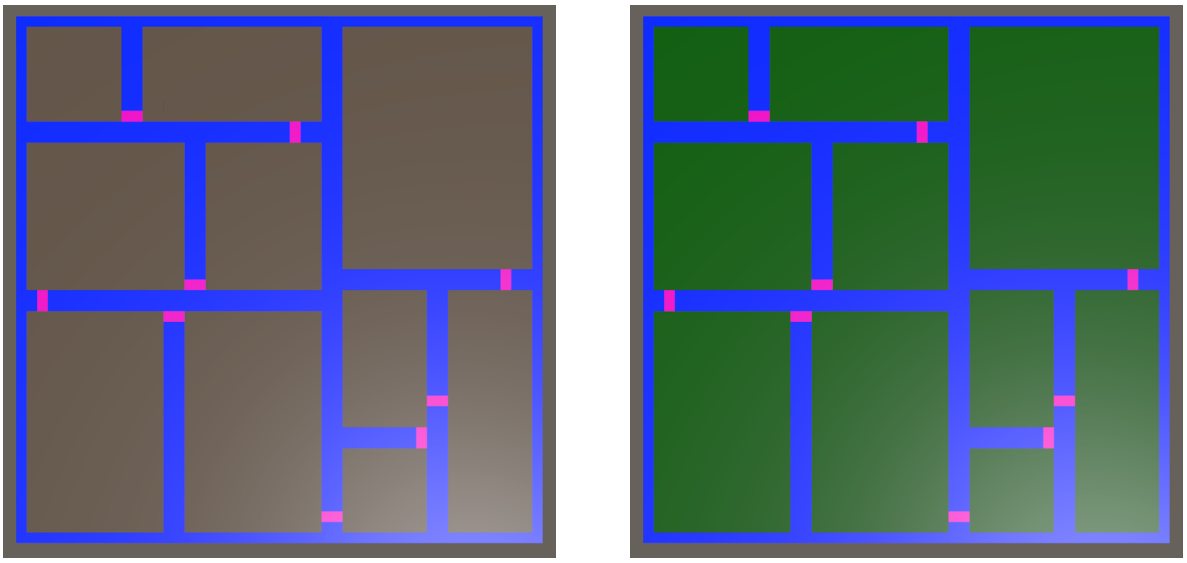
Зураг 5.1: Үр дүнгийн загвар

5.1.1 BSP

BSP-г тодорхойлноор төлөөллийн загварын буюу орчны өрөөнүүдийн ерөнхий бүтцийг гаргана. Өөр өнгийн тэгш өнцөгтүүд нь өөр өөр өрөөнүүдийг илтгэнэ 5.1a.

5.1.2 GridXZ-н GridCell-үүдэд Wall төрлийг оноох

Өрөө болгоны ханыг тодорхойлно. Цэнхэр хэсэг нь Wall төрөлтэй GridCell-г загварчилж байна 5.1b.



(a) Хана, хаалгуудыг тодорхойлсон GridXZ (b) Хана, хаалга, шалыг тодорхойлсон GridXZ

Зураг 5.2: Үр дүнгийн загвар

5.1.3 GridXZ-н GridCell-үүдэд Door төрлийг оноох

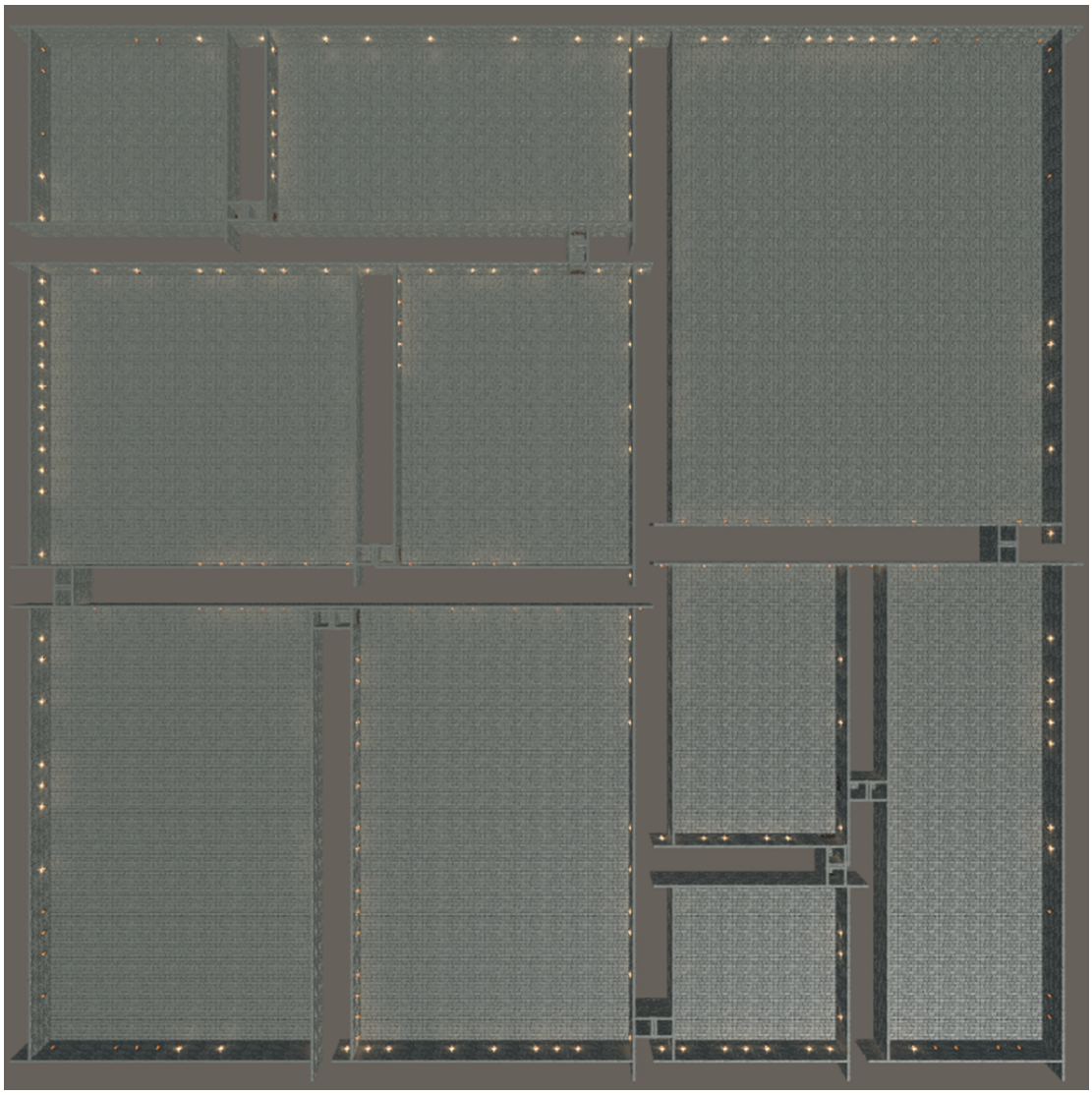
Өрөө болгоны хаалгыг тодорхойлно. BSP-г ашиглаж хаалганы оноолтыг хийх ба аль ч өрөөнөөс аль ч өрөө рүү шилжих боломжтой, өрөө болгох нь хоёр хаалгатай. Хаалганы нэгжийг ягаан өнгөөр дүрлэв 5.2a

5.1.4 GridXZ-н GridCell-үүдэд Floor төрлийг оноох

Орчны төлөөллийн загварын шалыг тодорхойлж байна. Ногоон өнгөөр дүрслэв 5.2b.

5.2 Тоглоомын моделийг үүсгэх

Төлөөллийн загварт суурилж тоглоомын орчинг үүсгэв 5.3.



Зураг 5.3: Тоглоомын орчин

5.3 CellUnit, CellBundle үүсгэх.

Хэрэглэгч эсвэл хөгжүүлэгч нь системийг ашиглахдаа шинээр CellBundle болон үүнд хамаарах CellUnit-уудыг үүсгэх боломжтой ба дараах байдлаар үүсгэнэ.

5.3.1 CellBundle үүсгэх

1. Шинээр *CellBundle* үүсгэнэ.

2. Өрөөний хамгийн жижиг байх хэмжээг *MIN_ROOM_SIZE*-д оноож өгнө.

3. Өрөөний хамгийн том байх хэмжээг *MAX_ROOM_SIZE*-д оноож өгнө.

4. BSP-н нүд мутацлах боломжийг *LEAF_NODE_CHANCE*-д оноож өгнө.

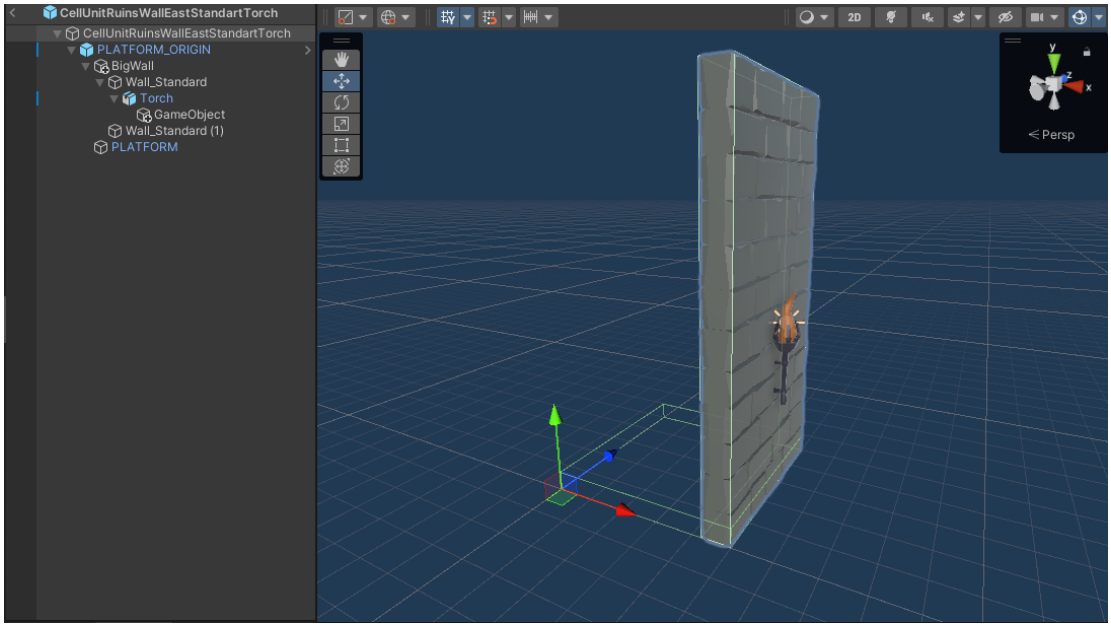
5. Тухайн *CellBundle*-н нэрийг Bundle Name-д оноож өнгө.

6. *Floor, Wall, Pillar, Abyss, Stair Down, Stair Up, Door* гэсэн жагсаалтын оролтын хэсэгт хэрэглэгч хамаарах *CellUnit*-уудыг оноож өгнө.

5.3.2 CellUnit үүсгэх

1. *PLATFORM_ORIGIN* prefab-г ашиглан *PLATFORM*-д харьяалагдах хэсгийг Зураг 5.4-н дагуу тодорхойлж өгнө.

2. *PLATFORM_ORIGIN* нь *PLATFORM* GameObject-г агуулсан prefab.



Зураг 5.4: CellUnit-г тодорхойлж буй жишээ

3. *PLATFORM* нь тухайн нэгжийн хязраарыг тодорхойлох collider component-тэй GameObject.

5.4 Хэрэглэгчийн тест, сайжруулалт.

Хэрэглэгчийн тест хийх явцад BSP-н навч биш нүд хоорондын коридорыг тодорхойлоход корридор нь өөр өрөөрүү биш хаалганы CellUnit гарах үр дүнг үүсгэх тохиолдол байсан. Энэ нь хүсэмжгүй үр дүн ба хаалга нь өөр өрөөрүү заавал харсан байх хэрэгтэй. Систем нь процедурт программчлалаар зохиогдсон учир орчны төлөөллийн загварын хаалгыг үүсгэх хэсэгт алдаа байгааг амархнаар илрүүлж чадсан.

5.4.1 Алдаа

Хаалганы байршил өрөөрүү бус корридорруу харах хэлбэрээр үүсэх шалтгаан нь навч өрөөнүүдийн хооронд хаалганы CellUnit-г үүсгэхдээ өөр корридорруу харахгүй байршилд үүсгэх дүрмийг тодорхойлж өгсөн ч эцэг нүд хоорондын хаалганы CellUnit-г тодорхойлохдоо үүнийг тодорхойлоогүй орхисон байв.

5.4.2 Засвар

Засварыг хийхдээ кодод Зураг 5.5-н дагуу өөрчлөлт оруулав. Дахин хэрэглэгчийн тестийг олон удаа хийхэд энэ алдаа гараагүй.

5.5 Хэлэлцүүлэг

Аливаа төвөгтэй процедурт үүсгэлтийг хэрэгжүүлэхдээ боловсруулалтыг үе шаттай хийвэл системийг өргөжүүлэх, өөрчлөлт оруулахад тустай байв. Өөрөөр хэлбэл функциональ хандлагаар олон модулиуд боловсруулалт дараалан хийнэ. Ингэхийн тулд хөгжүүлэгч модулиудын хоорондын харилцаа, оролт гаралтыг тодорхойлсон байх шаардлагатай ба хамгийн түрүүнд хөгжүүлэгч хүссэн үр дүнгээ тодорхойлсон байх ёстой. Аливаа процедурт үүсгэлтийн системийн доторх

```
private static void AssignDoorsForHorizontalCutSiblings(GridXZ<GridCell> gridXZ, Coord2D rightCoord,
    BinarySpaceTree leftChild)
{
    int randomX = Random.Range(rightCoord.X + 1, rightCoord.X + leftChild.NodeRectangle.Width - 1);
    var cellLeft = gridXZ.GetGridObject(randomX, rightCoord.Z - 1);
    var cellRight = gridXZ.GetGridObject(randomX, rightCoord.Z);

    while (cellLeft.CellProceduralGenerationUnitType == ProceduralGenerationUnitType.Door ||
           cellLeft.CellProceduralGenerationUnitType == ProceduralGenerationUnitType.Door)
    {
        randomX = Random.Range(rightCoord.Z + 1, rightCoord.Z + leftChild.NodeRectangle.Height - 1);
        cellLeft = gridXZ.GetGridObject(randomX, rightCoord.Z - 1);
        cellLeft.CellDirection = Direction.West;
        cellRight = gridXZ.GetGridObject(randomX, rightCoord.Z);
        cellRight.CellDirection = Direction.East;
    }

    cellLeft.CellProceduralGenerationUnitType = ProceduralGenerationUnitType.Door;
    cellRight.CellProceduralGenerationUnitType = ProceduralGenerationUnitType.Door;
}
```

```
136
137     private static void AssignDoorsForHorizontalCutSiblings(GridXZ<GridCell> gridXZ, Coord2D rightCoord,
138         BinarySpaceTree leftChild)
139     {
140         ChooseDoorCellsHorizontalCut(gridXZ, rightCoord, leftChild, out var cellLeft, out var cellRight);
141         cellLeft.CellDirection = Direction.South;
142         cellRight.CellDirection = Direction.North;
143         cellLeft.CellProceduralGenerationUnitType = ProceduralGenerationUnitType.Door;
144         cellRight.CellProceduralGenerationUnitType = ProceduralGenerationUnitType.Door;
145     }
146
147     private static void ChooseDoorCellsHorizontalCut(GridXZ<GridCell> gridXZ, Coord2D rightCoord,
148         BinarySpaceTree leftChild, out GridCell cellLeft,
149         out GridCell cellRight)
150     {
151         int randomX = Random.Range(rightCoord.X + 1, rightCoord.X + leftChild.NodeRectangle.Width - 1);
152
153         var southCellOfCellLeft = gridXZ.GetGridObject(randomX, rightCoord.Z - 2);
154         var northCellOfCellRight = gridXZ.GetGridObject(randomX, rightCoord.Z + 1);
155
156         while (southCellOfCellLeft.CellProceduralGenerationUnitType == ProceduralGenerationUnitType.Solid ||
157                northCellOfCellRight.CellProceduralGenerationUnitType == ProceduralGenerationUnitType.Solid)
158         {
159             randomX = Random.Range(rightCoord.X + 1, rightCoord.X + leftChild.NodeRectangle.Width - 1);
160
161             southCellOfCellLeft = gridXZ.GetGridObject(randomX, rightCoord.Z - 2);
162             northCellOfCellRight = gridXZ.GetGridObject(randomX, rightCoord.Z + 1);
163         }
164
165         southCellOfCellLeft.CellProceduralGenerationUnitType = ProceduralGenerationUnitType.Door;
166         northCellOfCellRight.CellProceduralGenerationUnitType = ProceduralGenerationUnitType.Door;
167         cellLeft = gridXZ.GetGridObject(randomX, rightCoord.Z - 1);
168         cellRight = gridXZ.GetGridObject(randomX, rightCoord.Z);
169     }
170
```

Зураг 5.5: Хаалга CellUnit-н алдааны засвар

модулиудын өөр олон төрлийг харин объект хандлагаар хэрэгжүүлбэл тоглоомын орчны олон талд байдлыг сайжруулахад илүү амар гэж таамаглаж байна.

5.5.1 Яагаад процедурт үүсгэлтийг тоглоомын явцад илүү ихээр хэрэгжүүлэхгүй байна бэ?

1. Тоглоомын явц нь тоглоомын томоохон гол хэсэг ба тухайн тоглоом сайн бүтээгдэхүүн болох эсэх дээр нөлөө их. Тоглоомын явцыг процедураар үүсгэхэд хөгжүүлэлтийн эхлэл хэсэгт бодит бүтээгдэхүүн харагдахгүй, оронд нь процедурт үүсгэлтийн системийн хөгжүүлэлт байх болно. Энэ нь бүтээгдэхүүнийг илүү нарийн төвөгтэй болгох санхүүжилт авахад хүндрэл учруулж болох юм.

2. Зарим тоглоомын төсөл нь PCG-н шинжүүдийг хүсэхгүй байж болно. PCG-с олох дахин тоглох чадвар үнэ цэнтэй ч энэ нь тоглоомын зах зээлд их ашиг олж чаддаггүй. Оронд нь үнэ цэнийг хэрэглэгчид дэвшилтэд технологи, график эсвэл гар аргаар хийхэд илүү тохиромжтой өвөрмөц сэтгэгдэл төрүүлэх тоглоомын явцуудаас хардаг.

3. Процедурт аргаар үүсгэсэн тоглоомын явц нь хоорондоо адилгүй эрс тэрс байж болно. Тоглогч нь тоглоомыг олон дахин тоглож байж бүх боломжит тоглоомын явцыг мэдрэх шаардлагатай байж магадгүй ба энэ нь цаг зав багатай хүнд сайн зүйл биш юм. Том компаниуд зорилтот хэрэглэгчдээ аль болох их байхыг хүснэ.

4. Мөн процедурт үүсгэлтийн систем нь хангалтгүй бол хөгжүүлэлтийн үйл явцыг хөнгөвчлөхийн оронд хөгжүүлэгчийн хүссэн аливаа загвар, зорилгод саад болох боломжтой.

5.5.2 Өөр ямар аргаар PCG-г тоглоомын явцад хэрэгжүүлбэл сайн бэ?

Энэ хэсэгт тоглоомын хувьд PCG-н сул талыг багасгах өнцгөөс шинжилгээ хийнэ. Тоглоомын үнэ цэнийг ихэсгэх, тоглогчдын сэтгэл ханамжийг сайжруулах зорилгоор аль хэдийн гар аргаар хийсэн тоглоомын явцтай тоглоомд процедурт үүсгэлтийн систем нэгтгэж дахин тоглох чанарыг сайжруулж болох юм. Ингэснээр бүх төрлийн тоглогчдыг татах боломжтой ба харьцангуй бага өртгөөр тоглогчдод нэмэлт үнэ цэнэ, сэтгэл ханамжийг өгөх боломжтой.

6. ДҮГНЭЛТ

PCG-г ашигласнаар тоглоомын хөгжүүлэлтийн явцыг хөнгөвчлөх, нөөцийг үр ашигтай ашиглах бүрэн боломжтой. Гэхдээ төлөвлөгөө, үр дүнгээ тодорхой болгож ямар систем яаж хийхээ сайн мэдэхгүй бол систем нь хангалтгүй болж цаашлаад тоглоомын хөгжүүлэлтийн явцад саад болох эрсдэлтэй. Сайн PCG-н системийн боловсруулахад хэцүү. Мөн зарим тоглоомын төсөлд PCG-г тоглоомын явцад хэрэгжүүлэх хүсэлгүй байж болох юм. Гэсэн хэдий ч PCG-г тоглоомын явцад оруулж өгсөн амжилттай жишээнүүд олон байдаг. PCG-н системийг хэрэгжүүлэх аргад суралцан, туршилт хийснээр уг чиглэлээр илүү сайжрах боломжтой ба PCG-н системийг илүү оновчтой хэрэгжүүлж болон өөр олон аргатай хослуулан өвөрмөц сайн бүтээлийг бага өртгөөр гаргаж болно.

Талархал

Судалгааны ажлын явцад мэдлэг болон судалгаа хийх боломжоор хангаж удирдсан Проф. Х. Оюундолгор багшдаа баярлалаа.

Bibliography

[1] Lawrence Johnson, Georgios Yannakakis, and Julian Togelius. "Cellular automata for real-time generation of". In: (Sept. 2010). DOI: 10.1145/1814256.1814266.

[2] Noor Shaker, Julian Togelius, and Mark J. Nelson. *Procedural Content Generation in Games*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 3319427148.

[3] Julian Togelius et al. "What is Procedural Content Generation? Mario on the borderline". In: (June 2011). DOI: 10.1145/2000919.2000922.

[4] Herbert Wolverson. *Procedural Map Generation Techniques*. https://youtu.be/TlLIOgWYVpI. 2020.

A. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

A.1 Системийн хэсэг

A.1.1 BinarySpaceTree

```
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using CodeMonkey.Utils;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.Rendering.UI;
using Random = UnityEngine.Random;


namespace DefaultNamespace
{
    public class BinarySpaceTree
    {
        public static int VERTICAL_CUT_INDEX = 0;
        public static int HORIZONTAL_CUT_INDEX = 1;

        private Rectangle _nodeRectangle;
        private BinarySpaceTree _leftChild;
        private BinarySpaceTree _rightChild;
        private BinarySpaceTree _parentCell;
        private bool _isLeaf = true;
        private int _level = 0;
        private ProceduralGenerationCellBundle _bundle;
        public int Key;

        public Rectangle NodeRectangle
        {
            get => _nodeRectangle;
            set => _nodeRectangle = value;
        }


        public bool IsLeaf => _isLeaf;

        public int Level => _level;

        public BinarySpaceTree RightChild
        {
            get => _rightChild;
            set
            {
                if (value == null)
```

```
                    return;
                _rightChild = value;
                _isLeaf = false;
            }
        }

        public BinarySpaceTree LeftChild
        {
            get => _leftChild;
            set
            {
                if (value == null)
                    return;
                _leftChild = value;
                _isLeaf = false;
            }
        }

        public BinarySpaceTree ParentCell
        {
            get => _parentCell;
            set
            {
                if (value == null)
                {
                    _parentCell = null;
                    _level = 0;
                }
                else
                {
                    _parentCell = value;
                    _level = _parentCell._level + 1;
                }
            }
        }


        public BinarySpaceTree(Rectangle nodeRectangle,
            ProceduralGenerationCellBundle bundle)
        {
            NodeRectangle = nodeRectangle;
            _bundle = bundle;
            ParentCell = null;
            Key = 1;
            Divide();
        }

        private BinarySpaceTree(Rectangle nodeRectangle,
            BinarySpaceTree parentCell,
            ProceduralGenerationCellBundle bundle, int key)
        {
            NodeRectangle = nodeRectangle;
```

```
            ParentCell = parentCell;
            _bundle = bundle;
            Key = key;
            Divide();
        }

        public void Divide()
        {
            if (CanMutateToBeBig()) return;
            int direction = GetDirectionToCut();
            if (direction == -1) return;
            CreateChildNodes(direction);
        }

        private void CreateChildNodes(int direction)
        {
            if (direction == VERTICAL_CUT_INDEX)
            {
                CreateChildNodesVertically();
            }
            else
            {
                CreateChildNodesHorizontally();
            }
        }

        private void CreateChildNodesHorizontally()
        {
            var divisionAmount = Random.Range(_bundle.MIN_ROOM_SIZE,
               NodeRectangle.Height + 1 - _bundle.MIN_ROOM_SIZE);
            divisionAmount = Math.Max(_bundle.MIN_ROOM_SIZE,
               divisionAmount);

            var leftChildCoords2D = new Coord2D(NodeRectangle.
               StartingCoord2D.X, NodeRectangle.StartingCoord2D.Z);
            var leftChildRectangle = new Rectangle(leftChildCoords2D,
               NodeRectangle.Width, divisionAmount);
            LeftChild = new BinarySpaceTree(leftChildRectangle, this,
               _bundle, Key * 2);

            var rightChildCoords2D = new Coord2D(NodeRectangle.
               StartingCoord2D.X,
                NodeRectangle.StartingCoord2D.Z + divisionAmount);
            var rightChildRectangle = new Rectangle(rightChildCoords2D,
                NodeRectangle.Width,
                NodeRectangle.Height - divisionAmount);
            RightChild = new BinarySpaceTree(rightChildRectangle, this,
                _bundle, Key * 2 + 1);
        }

        private void CreateChildNodesVertically()
        {
```

```
            var divisionAmount = Random.Range(_bundle.MIN_ROOM_SIZE, NodeRectangle.Width + 1 - _bundle.MIN_ROOM_SIZE);
            divisionAmount = Math.Max(_bundle.MIN_ROOM_SIZE, divisionAmount);

            var leftChildCoords2D = new Coord2D(NodeRectangle.StartingCoord2D.X, NodeRectangle.StartingCoord2D.Z);
            var leftChildRectangle = new Rectangle(leftChildCoords2D, divisionAmount, NodeRectangle.Height);
            LeftChild = new BinarySpaceTree(leftChildRectangle, this, _bundle, Key * 2);

            var rightChildCoords2D = new Coord2D(NodeRectangle.StartingCoord2D.X + divisionAmount,
                NodeRectangle.StartingCoord2D.Z);
            var rightChildRectangle = new Rectangle(rightChildCoords2D,
                NodeRectangle.Width - divisionAmount,
                NodeRectangle.Height);
            RightChild = new BinarySpaceTree(rightChildRectangle, this,
                _bundle, Key * 2 + 1);
        }

        private int GetDirectionToCut()
        {
            int direction;
            if (NodeRectangle.Width >= _bundle.MIN_ROOM_SIZE * 2 && NodeRectangle.Height < _bundle.MIN_ROOM_SIZE * 2)
                direction = VERTICAL_CUT_INDEX;
            else if (NodeRectangle.Width < _bundle.MIN_ROOM_SIZE * 2 &&
                     NodeRectangle.Height >= _bundle.MIN_ROOM_SIZE * 2)
                direction = HORIZONTAL_CUT_INDEX;
            else if (NodeRectangle.Width >= _bundle.MIN_ROOM_SIZE * 2 &&
                     NodeRectangle.Height >= _bundle.MIN_ROOM_SIZE * 2)
                direction = Random.Range(0, 2);
            else
                return -1;
            return direction;
        }

        private bool CanMutateToBeBig()
        {
            if (_level == 0) return false;
            if (NodeRectangle.Width < _bundle.MAX_ROOM_SIZE && NodeRectangle.Height < _bundle.MAX_ROOM_SIZE)
            {
                // multiplying because the generator has 0.5 chance multiplier
                // max bundle leaf node input is 50, so max is 50%
                if (RandomUtils.Chance(_bundle.LEAF_NODE_CHANCE * 2))
                    return true;
            }
```

```
            return false;
        }

        public void AssignNodesAtLevelFromRootNode(int level, List<
            BinarySpaceTree> nodesAtLevel)
        {
            if (Key != 1)
                throw new InvalidOperationException("This is not the
                    root node");
            AssignNodesAtLevelForNode(level, nodesAtLevel);
        }

        private void AssignNodesAtLevelForNode(int level, List<
            BinarySpaceTree> nodesAtLevel)
        {
            if (Level == level)
            {
                nodesAtLevel.Add(this);
            }

            if (LeftChild != null)
                LeftChild.AssignNodesAtLevelForNode(level, nodesAtLevel
                    );
            if (RightChild != null)
                RightChild.AssignNodesAtLevelForNode(level,
                    nodesAtLevel);
        }


        public void AssignLeafNodesFromRootNode(List<BinarySpaceTree>
            leafNodes)
        {
            if (Key != 1)
                throw new InvalidOperationException("This is not the
                    root node");
            AssignLeafNodes(leafNodes);
        }

        private void AssignLeafNodes(List<BinarySpaceTree> leafNodes)
        {
            if (IsLeaf)
            {
                leafNodes.Add(this);
                return;
            }

            LeftChild.AssignLeafNodes(leafNodes);
            RightChild.AssignLeafNodes(leafNodes);
        }

        public BinarySpaceTree GetSibling()
```

```
        {
            if (ParentCell == null)
                throw new InvalidOperationException("This node doesn't
                    have a parent");
            if (Key != ParentCell.LeftChild.Key)
                return ParentCell.LeftChild;

            return ParentCell.RightChild;
        }

        public bool IsRightChild()
        {
            if (ParentCell == null)
                throw new InvalidOperationException("This node doesn't
                    have a parent");
            if (ParentCell.RightChild.Key == Key)
            {
                return true;
            }

            return false;
        }

        public static int GetLevelDepth(BinarySpaceTree binarySpaceTree
            , int initialLevel = 1)

        {
            if (binarySpaceTree.Key != 1)
                throw new InvalidOperationException("This is not the
                    root node");

            List<BinarySpaceTree> nodesAtTheBottom = new List<
                BinarySpaceTree>();
            binarySpaceTree.AssignLeafNodes(nodesAtTheBottom);

            nodesAtTheBottom.ForEach(e =>
            {
                if (e.Level > initialLevel)
                {
                    initialLevel = e.Level;
                }
            });
            return initialLevel;
        }


        public BinarySpaceTree GetNodesAtIndex(int index)
        {
            if (this.Key != 1)
                throw new InvalidOperationException("This is not the
                    root node");
            return GetNodeAtIndexHelper(this, index);
```

```
        }

        // in-order traversal
        private static BinarySpaceTree GetNodeAtIndexHelper(
            BinarySpaceTree node, int index)
        {
            if (node == null)
                return null;

            if (node.Key == index)
                return node;
            BinarySpaceTree leftNode = GetNodeAtIndexHelper(node.
                LeftChild, index);

            if (leftNode != null)
            {
                if (leftNode.Key == index)
                    return leftNode;
            }

            return GetNodeAtIndexHelper(node.RightChild, index);
        }

        public static void InorderTraversalDebugLog(BinarySpaceTree
            root)
        {
            if (root != null)
            {
                InorderTraversalDebugLog(root.LeftChild);
                Debug.Log(root.Key + " ");
                InorderTraversalDebugLog(root.RightChild);
            }
        }

        public Coord2D GetStartingCoords()
        {
            return NodeRectangle.StartingCoord2D;
        }
    }
}
```

Код А.1: BinarySpaceTree хэрэгжүүлэлт

В. АЖЛЫН ТӨЛӨВЛӨГӨӨ

2023 оны 02 сарын 06 нд

Монгол нэр: Дүрмээр тодорхойлогдох график загварчлалын арга түүний хэрэглээ
(англи нэр): Rule-based procedural modeling
Сэдэвт бакалаврын судалгааны ажлын
7 хоногийн үечилсэн төлөвлөгөө

Хугацаа: **2022.02.06**-аас **2022.05.08** хүртэл **13** долоо хоног

№	**Хийх ажил**	**Долоо хоног**	**1**	**2**	**3**	**4 Явц I**	**5**	**6**	**7**	**8**	**9 Явц II**	**10**	**11**	**12 Урьдчил - сан хамгаа-лалт**	**13**	**14 Жинхэ - нэ хамгаа-лалт**	**Тайлбар**
1	**Судалгаа**	Сэдвийн судалгаа															
		Технологийн судалгаа															
		Алгоритмын судалгаа															
2	**Хэрэгжүүлэлт**	Хэрэглэгчийн харилцах хэсгийн зохиомж															
		Алгоритмын хэрэгжүүлэлт															
		Үр дүнгүүдийг харьцуулах															
3	**Сайжруулалт**	Хэрэглэгчийн тест хийн, өөрчлөлтийг тодорхойлох															
		Өөрчлөлтийг хийн сайжруулах															
4	**Баримт бичиг**	Төслийн баримт бичиг															
		Баримт бичгийг цэгцлэн тайланг бэлэн болгох															

Зураг B.1: Ажлын төлөвлөгөө