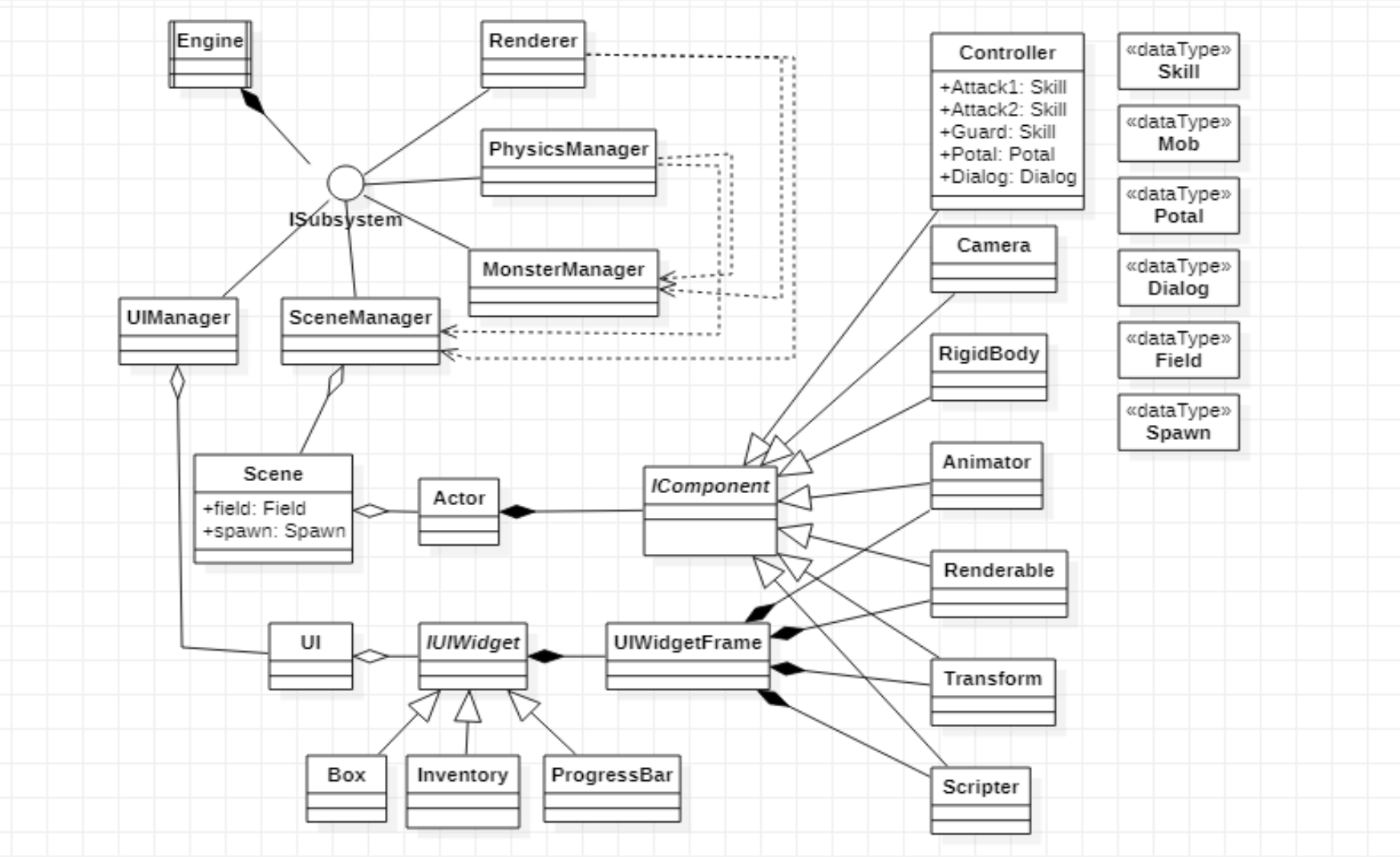
DirectX 2D MiniGame

https://mona04.github.io/posts/portfolio/DirectX-2D

https://github.com/Mona04/MiniGame-DirectX-2D

오경민, Kyung-Min Oh



01 / Physics

기능
Update, 충돌 검사
충돌 처리

02 / Battle

Pooling, AI

03 / Rendering

Sprite
Instancing

04 / Save Load

Load

05 / UI

Widget
Picking
Dialog
Inventory

-The above Image is linked to YouTube

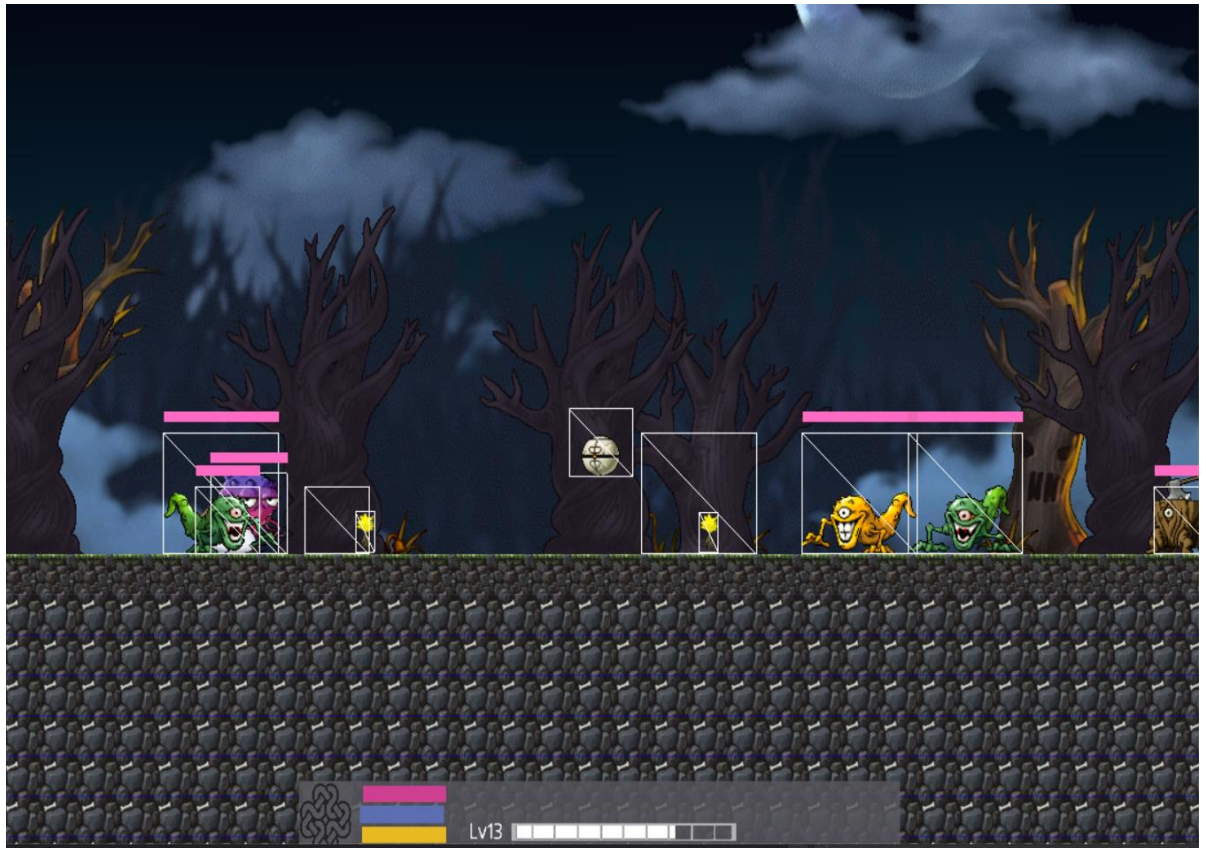
https://mona04.github.io/posts/portfolio/DirectX-2D

https://github.com/Mona04/MiniGame-DirectX-2D

오경민, Kyung-Min Oh

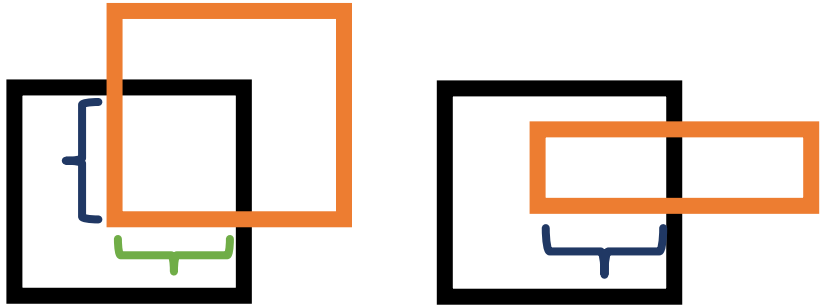
PhysicsManager

- 회전이 없는 Sprite 게임이므로 간단한 직사각형 충돌검사가 기본이 됩니다.
- RigidBody 는 Movable, Unmovable, Overlapped 등으로 타입을 지정할 수 있고, 특정 Type 만 물리적 효과를 주었습니다.
- Trace 기능을 만들어, 아이템 줍기나 NPC 대화 시에 충돌탐지를 할 수 있게 했습니다.
- 지형 블록의 경우 배열에 값을 넣어 x, y 위치로 직접접근이 가능하도록 하여 다른 Rigidbody 들과 빠른 충돌계산을 가능하게 했습니다.



- 몬스터, 아이템 등의 Boundbox 영역이 사각형 프레임으로 보이고 있다.

https://mona04.github.io/posts/portfolio/DirectX-2D
https://github.com/Mona04/MiniGame-DirectX-2D
오경민, Kyung-Min Oh



- 오른쪽과 달리 왼쪽은 하단충돌로도 왼쪽 충돌로도 간주할 수 있다.
- 기준이 있기 때문에 파란 괄호만큼 더하는 것으로 결정된다.

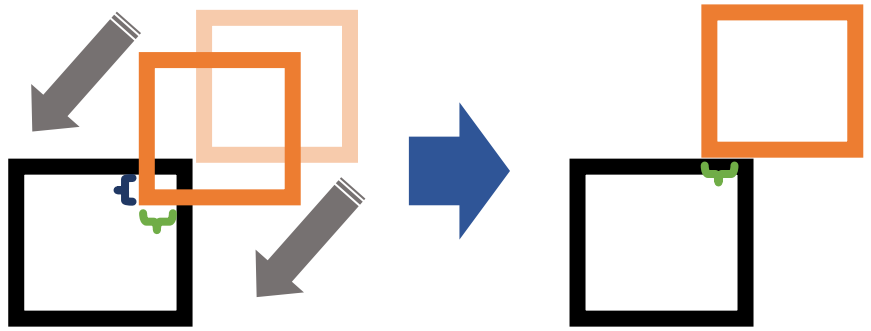
충돌검사

- 어떤 사각형이 상하좌우 중 어디 쪽에 부딪혔는지 확인하기 위해 상하좌우로 처리하는 기준이 필요합니다.
  - 움직이기 전의 원래위치의 Bottom 이 검사대상인 Top 보다 작으면 좌우가 아니라 하단 충돌로 간주합니다.
  - 상단 충돌도 마찬가지로 처리합니다.
- 위의 절차 상 상하 충돌이 우선으로 판단되므로 오른쪽 그림 같은 경우 상식적인 결과를 얻을 수 있습니다.

Physics Update 순서

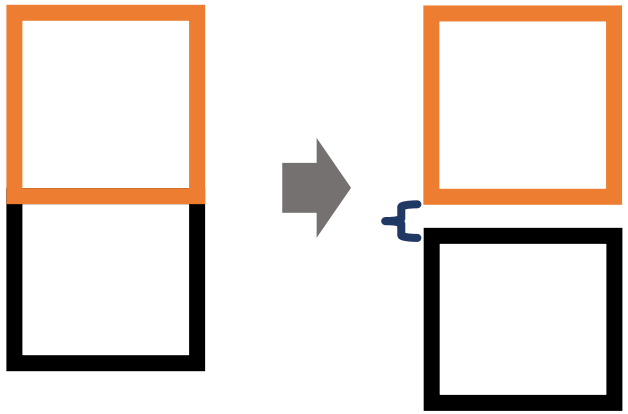
- Actor 를 Controller 로 이동시
  - ➢ RigidBody 의 MoveVector 와 Velocity 가 변경됩니다.
- 이러한 RigidBody 를 Tick 마다 아래 4가지 단계를 거쳐 업데이트 합니다.

1. 중력처리를 위해 Velocity 에 추가 값을 더합니다.
2. MoveVector 와 Velocity 를 사용해 Boundbox 를 업데이트합니다.
3. 충돌검사를 진행해 겹치는 차만큼 Boundbox 를 다시 업데이트 합니다.
4. 계산된 Boundbox 의 결과로 최종 위치를 확정 짓습니다.



- 위와 같은 하단/왼쪽 충돌 모두 가능한 상황을 상단에 우선순위를 두어 해결하였다.

https://mona04.github.io/posts/portfolio/DirectX-2D
https://github.com/Mona04/MiniGame-DirectX-2D
오경민, Kyung-Min Oh



- 충돌 처리시에 간격을 두어 원활한 계산을 가능하게 한다.

Chink

- 면과 면이 정확히 맞닿아 있으면 부동소수점 오차로 문제가 발생합니다.
  - 충돌처리 이후에도 충돌한 채로 남아있을 수 있고
  - 이전의 위치를 이용한 상하/좌우 충돌 판단에 오류가 생길 수 있습니다.

- 이를 위해서 충돌처리 때 미세한 틈(Chink)을 더 만듭니다.
- Chink 가 크면 부들거리는 것이 눈에 보이기 때문에 0.1 정도의 미세한 값을 사용했습니다.

좌우처리 우선

- 세로로 긴 블록을 향해 점프하면 가로 경계에서 점프가 막히는 현상이 생겼습니다.
- 이는 충돌 검사 시 상하충돌을 우선으로 한 것이 원인이었습니다.
  - 점프 중 블록의 Bottom 보다 캐릭터의 원래 Top 이 커 상단충돌 판정
- 모든 충돌체에 대해 좌우충돌처리 후 상하충돌처리를 하여 해결했습니다.
  - 좌우로 먼저 밀어내기 때문에 벽에서 상하충돌이 성립하지 않습니다.
  - 판단 기준은 상하가 우선, 처리는 좌우가 우선

- 오른쪽 그림처럼 블록이 세로로 있을 때 상단충돌이 일어나면 안된다.

https://mona04.github.io/posts/portfolio/DirectX-2D
https://github.com/Mona04/MiniGame-DirectX-2D
오경민, Kyung-Min Oh

Pooling

- Monster 와 Skill Effect 는 Pooling 패턴을 적용해 메모리 단편화를 없애고 빠른 탐색을 가능하게 했습니다.

AI

- 앞에서 본 PhysicsManager 의 지형 Trace 기능으로 몬스터가 계단에서 점프를 하고 큰 절벽 너머로는 가지 않도록 했습니다.
- 적대적 AI 는 적의 개체를 저장해, 거리 유지 / 접근 및 공격 두가지 상태를 일정 기간을 두고 전환합니다.

- 몬스터의 AI 가 Player 에게 스킬을 사용하는 모습이다.

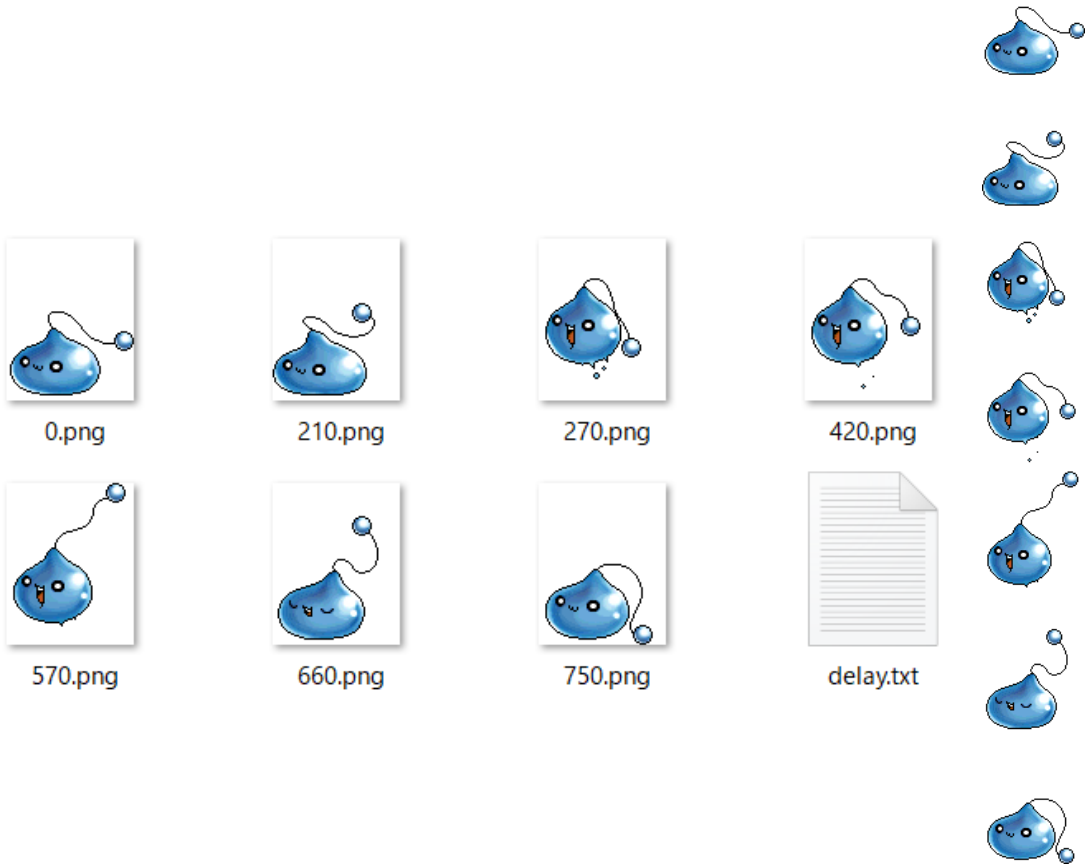
https://mona04.github.io/posts/portfolio/DirectX-2D
https://github.com/Mona04/MiniGame-DirectX-2D
오경민, Kyung-Min Oh

Sprite

- 단편적인 이미지를 하나의 Sprite Image 로 만들어 사용했습니다.
- (input.uv * SpriteSize + SpriteOffset) / TextureSize 식을 통해 Sprite Image 의 Frame 에 대한 UV 를 구했습니다.
- 바라보는 방향 정보를 따로 저장해, 1-uv.x 를 하여 이미지의 좌우를 뒤집었습니다.

Keyframe Update

- 누적시간을 구하고 Frame 간의 시간차가 넘으면
- Cur_Keyframe % Max_KeyFrame, acc_time -= time_interval 을 하여 Keyframe 을 재생했습니다.



- Image 소스를 하나의 Sprite Image 로 합치고, Delay 정보를 따로 파일로 만들어 보관하였다.

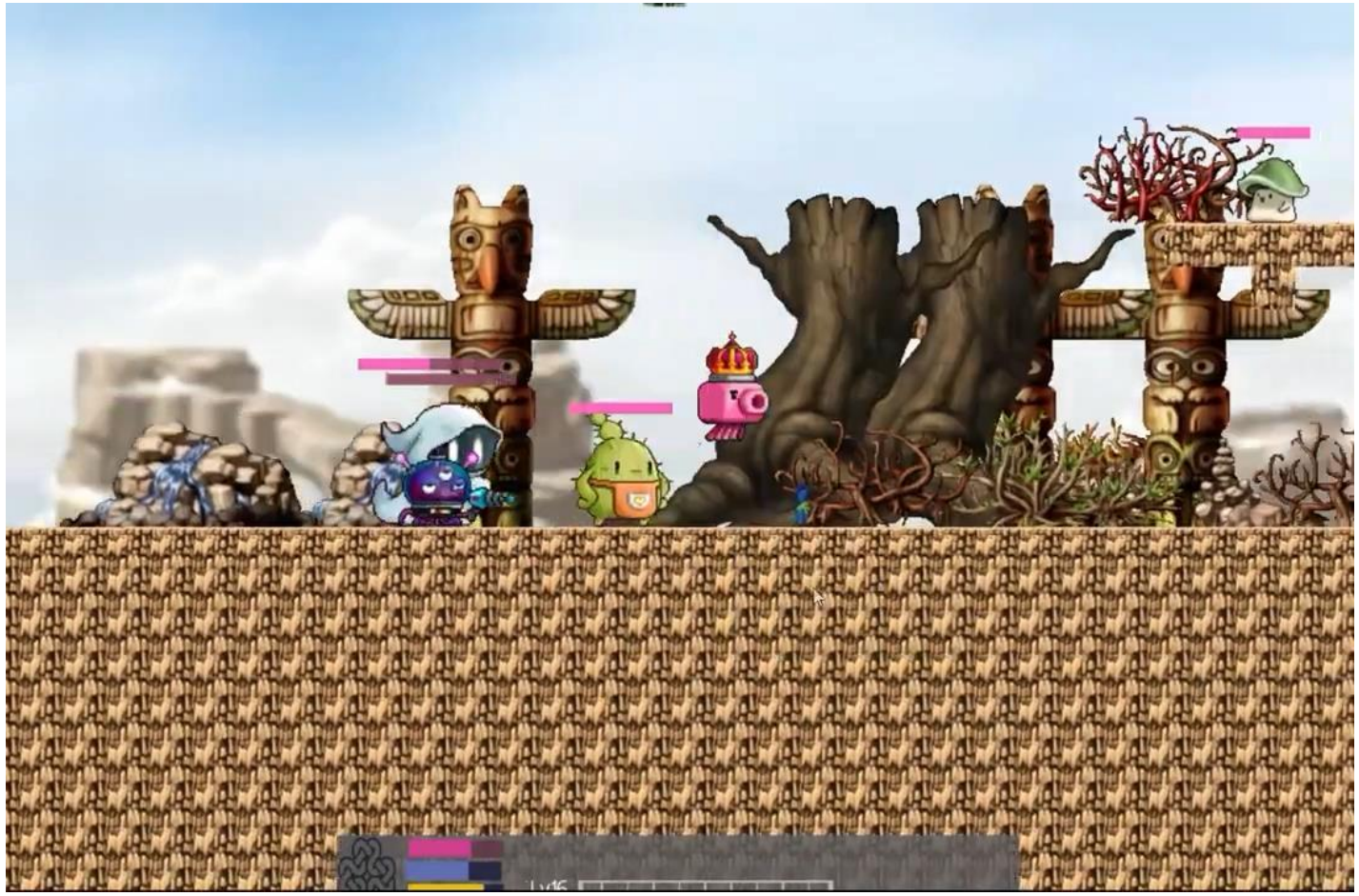
https://mona04.github.io/posts/portfolio/DirectX-2D

https://github.com/Mona04/MiniGame-DirectX-2D

오경민, Kyung-Min Oh

Instancing

- 지형블록, 환경블록 등은 매우 많은 개수가 화면에 그려지지만, 위치만 다르고 크기와 텍스쳐 등을 공유합니다.
- 따라서 DirectX 의 Instancing 기능을 이용해 DrawCall 을 블록의 종류 수만큼으로 줄였습니다.

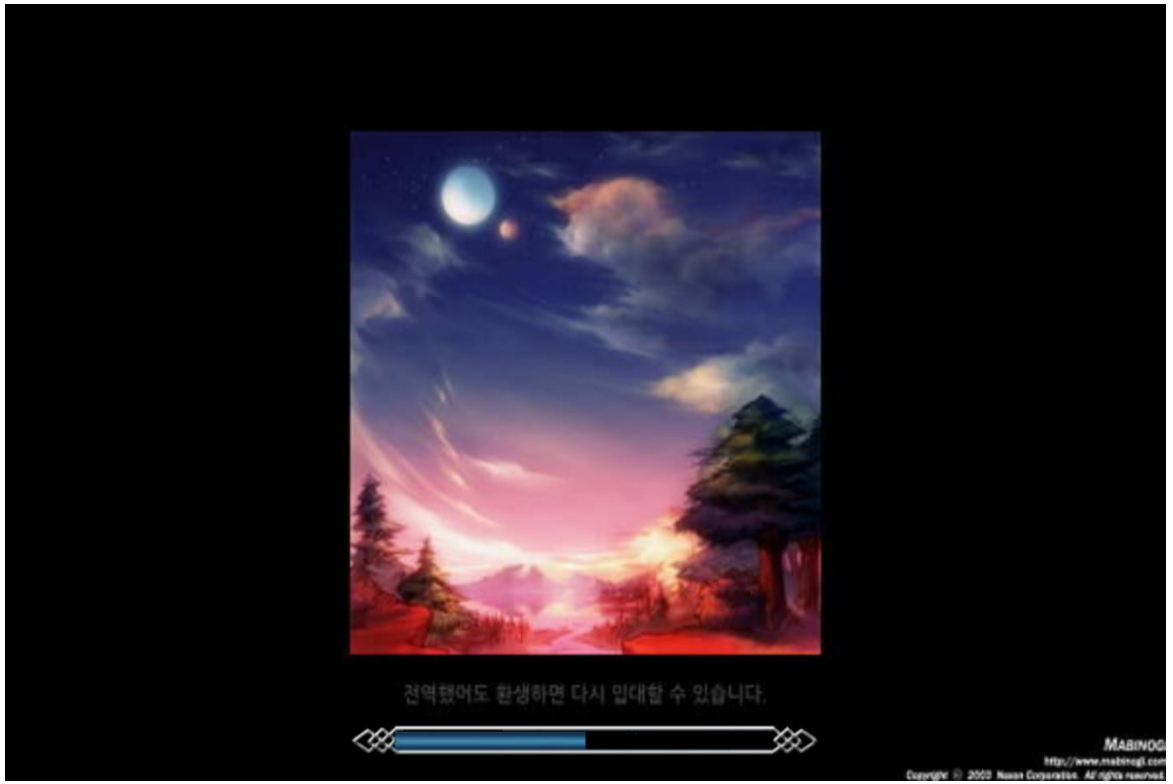


- 몇 백개가 넘어가는 작은 블록을 Instancing 으로 빠르게 그린다.

https://mona04.github.io/posts/portfolio/DirectX-2D
https://github.com/Mona04/MiniGame-DirectX-2D
오경민, Kyung-Min Oh

Loading

- Load 작업은 Thread Pool 을 꺼내 수행시켰습니다.
- ProgressBar 와 연동되는 ProgressReporter 를 만들어, 일정 작업이 끝나면 게이지가 차도록 명령을 넣었습니다.
  - Thread 안정성을 위해 ProgressReporter 의 메소드에는 Mutex 로 Lock 처리를 했습니다.
- ProgressReporter 로 Loading 중으로 확인 시
  - Update 를 하지 않고 Loading UI 만 그려서 Data Race 상태에 걸리지 않게 했습니다.



- Scene 전환 시에 나타나는 Loading 화면이다.

https://mona04.github.io/posts/portfolio/DirectX-2D

https://github.com/Mona04/MiniGame-DirectX-2D

오경민, Kyung-Min Oh

Widget

- Widget 은 사각형인 Frame 으로 구성되며, 전반적인 기능이 구현되어 있습니다.
  - Box, ProgressBar 처럼 기본적 Widget
  - 진화, 저장, 인벤토리, 툴팁의 특수 Widget
- Frame 은 Script 을 가지기 때문에 Frame 단위의 동작을 가능하게 했습니다.
- 한글은 Freetype Lib 을 사용해 렌더링 했습니다.
  - 글꼴 텍스쳐에 대한 Vertex / Index 의 변경

- 게임에서 사용되는 다양한 Widget 들이다.

https://mona04.github.io/posts/portfolio/DirectX-2D

https://github.com/Mona04/MiniGame-DirectX-2D

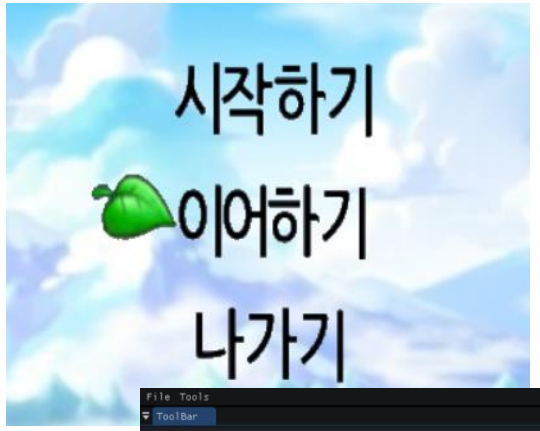
오경민, Kyung-Min Oh

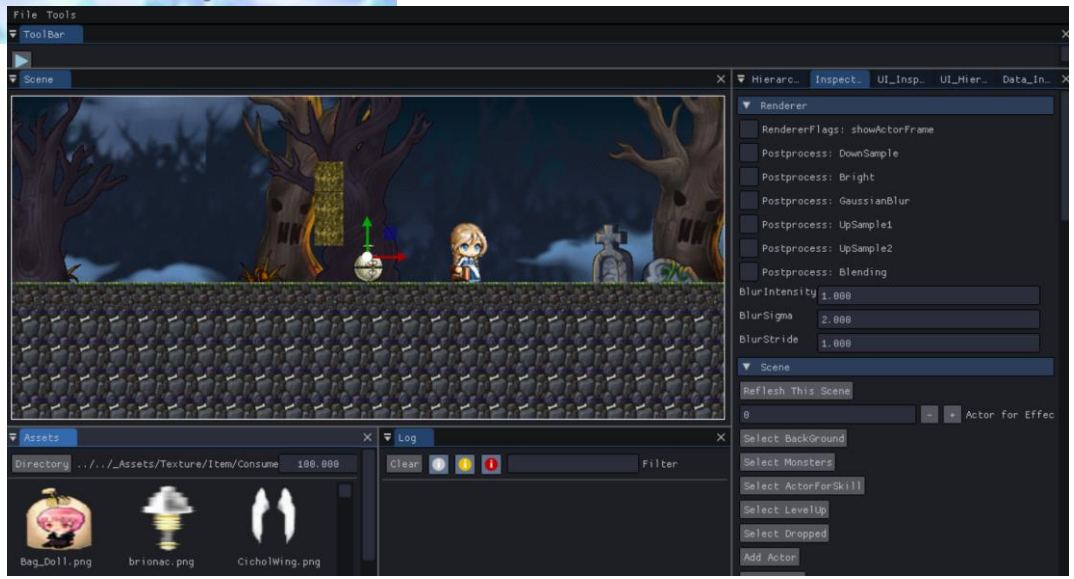
UI Picking

- UI 는 NDC 좌표를 직접 사용했습니다
- 윈도우 기준인 마우스 위치를 해상도로 나누어 정규화 시킴으로써 NDC 로 간단히 변환했습니다.
- 매 Tick 마다 각 Widget Frame 은 정규화된 마우스 위치와 겹치는지 확인해서 상태를 저장했습니다.
- 저장된 상태를 사용해 Script 등에서 원하는 동작을 수행합니다.

Scene Picking

- Ortho Projection 을 사용하므로 NDC 상의 마우스좌표를 ViewProj 의 역행렬을 사용해 변환시키면 마우스의 World 좌표를 얻을 수 있습니다.
- 이 좌표와 Actor 의 충돌을 검사하는 방식으로 간략히 Actor Picking 을 구현했습니다.





- (위) 마우스 위치에 맞추어 메뉴에 잎사귀로 포커스가 표시된다.
- (아래) Editor Viewport 에서 Actor 의 위치를 클릭 시 선택된다.

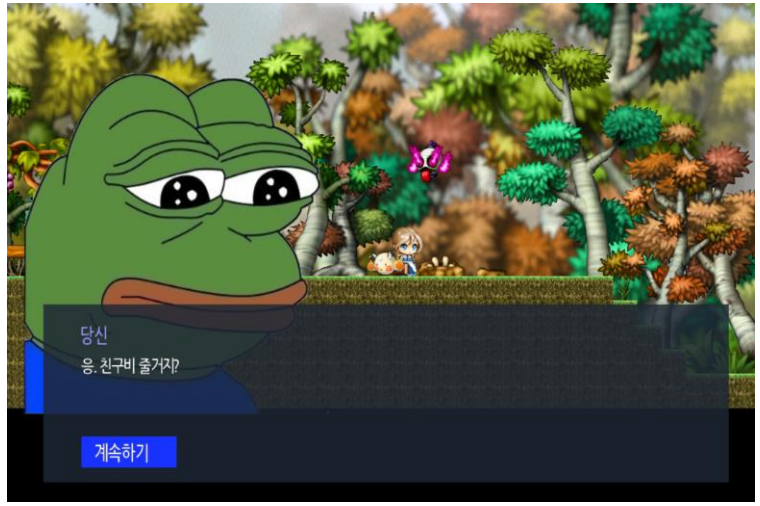
https://mona04.github.io/posts/portfolio/DirectX-2D

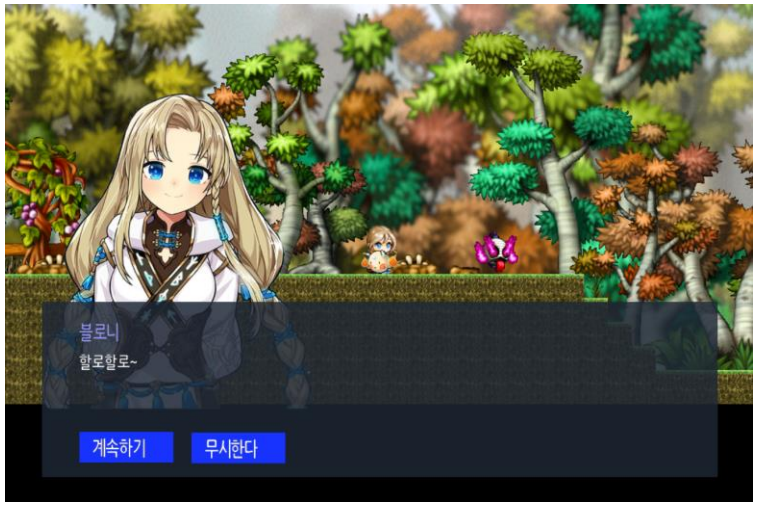
https://github.com/Mona04/MiniGame-DirectX-2D

오경민, Kyung-Min Oh

Dialog

- Dialog UI 에 기본적 Layout 을 만들고
- Dialog Manager 에서 Script 에서 사용할 함수를 만들어
- Script 에서 대화, 퀘스트 등을 구현했습니다.






```
switch(_cur)
{
case 0:
{
    dialogManager.DialogDefaultUpdate();
    dialogManager.DialogSetIllust("UI/NPC/Blony.png");
    dialogManager.DialogSetName("블로니");
    dialogManager.DialogSetText("할로할로~");
    dialogManager.DialogSetButton(0, " 계속하기");
    dialogManager.DialogSetButton(1, " 무시한다");
    _cur += 1;
    break;
}
case 1:
{
    if(dialogManager.IsDialogButtonClicked(0))
        _cur += 1;
    if(dialogManager.IsDialogButtonClicked(1))
    {
        _cur = 200;
    }
    break;
}
case 2:
```

- 위의 스크립트가 오른쪽의 화면을 만든다..

https://mona04.github.io/posts/portfolio/DirectX-2D

https://github.com/Mona04/MiniGame-DirectX-2D

오경민, Kyung-Min Oh

Inventory

- Data 와 Widget 을 분리하여 구현했습니다.
- 아이템은 인벤토리에서 차지하는 크기가 있습니다
  - Data 에서는 LeftTop 에만 정보를 저장하고 나머지는 Occupied 상태로 만들었습니다.
  - Widget 에서는 LeftTop 만 이미지를 그리고, 이미지의 사이즈가 차지하는 Slot 의 크기가 되도록 했습니다.
- n 개의 넣기, 1개 사용, 1개 드래그 및 분리를 구현했습니다.
- 아이템 위에 커서가 있으면 Inventory Tooltip 에 데이터를 주어 일정시간이 지나면 Tooltip 이 보이게 했습니다.

- 아이템 위에 커서를 두어 Tooltip 이 보이는 중이다.