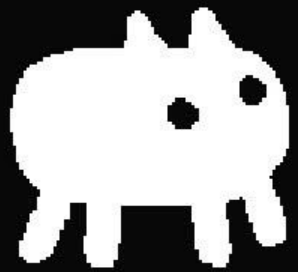
BA BA IS YOU



이현중

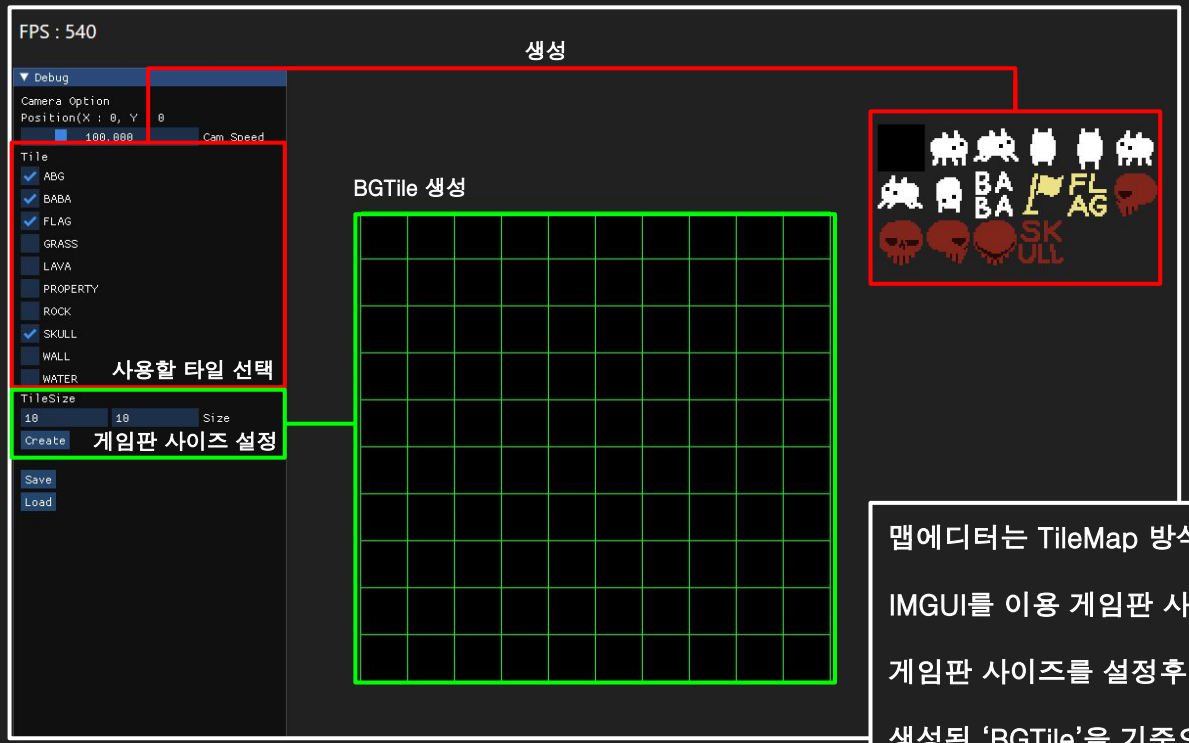
게임 개요

Vidio - https://youtu.be/rqB7eCFoilU

타이틀	BABA IS YOU 모작
장르	Puzzle
플렛폼	PC
라이브러리	WinApi, FMOD, IMGUI, DirectTex
개발언어	C++
개발 기간	1개월
개발 인원	1명

MapEditer – TileMap

맵에디터 화면



타일 배치 함수

```
// 선택한 타일을 기존BG타일 위치에 생성
void BabaTileMap::ClickTile(Object* selectSample)
{
    DeleteObjTile();
    if (KEY_PRESS(VK_LBUTTON))
    {
        for (Object* bgTile : bgTiles)
        {
            if (bgTile->GetCollider()->IsPointCollision(mousePos))
            {
                Object* object = new Object(selectSample->tag);
                object->Position() = bgTile->Position();
                object->SetParent(this);
                object->UpdateTransform();
                tiles.push_back(object);
                if (instanceQuads.count(object->tag) == 0)
                    instanceQuads[object->tag] = new InstanceQuad(object->t
                instanceQuads[object->tag]->AddPushDatas(object);

                for (Object* tile : instanceQuads[object->tag]->GetObjects(
                {
                    tile->GetInstanceData().curFrame.y = 0;
```

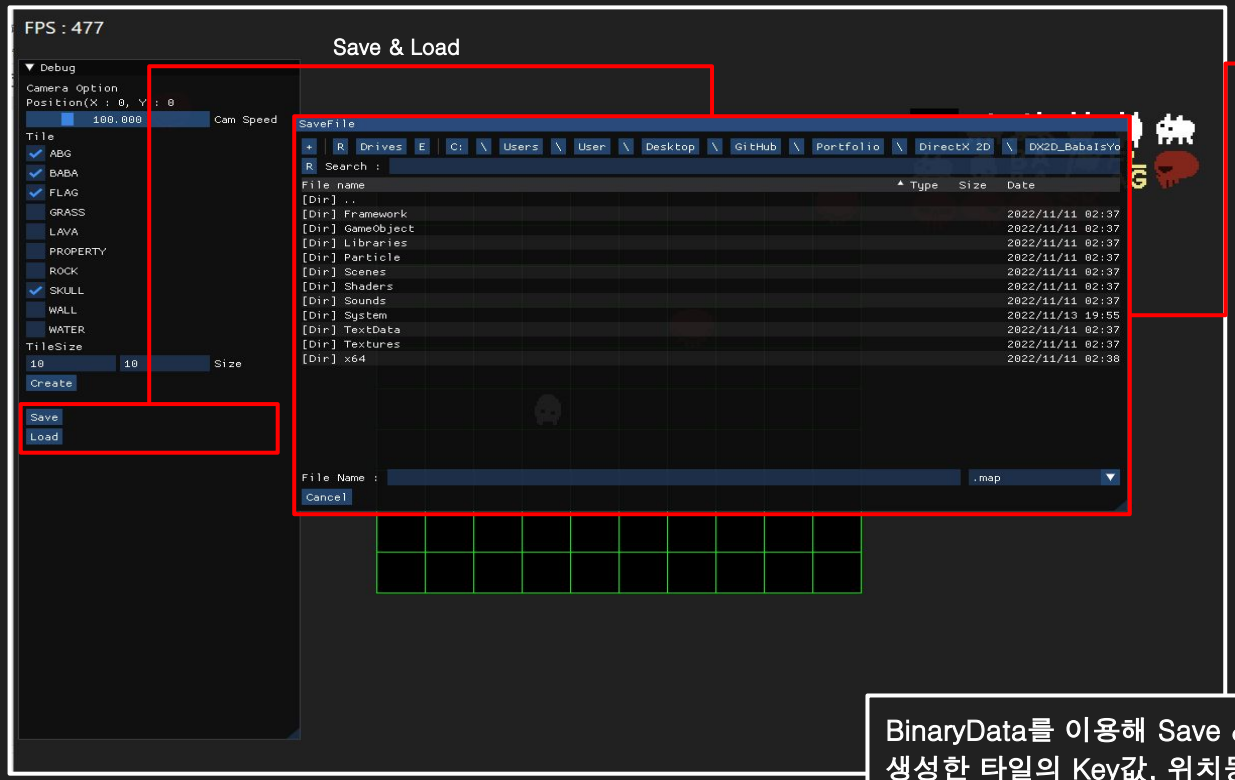
맵에디터는 TileMap 방식을 이용

IMGUI를 이용 게임판 사이즈와 사용할 Object를 가시화 할수있도록 제작

게임판 사이즈를 설정후 바탕이되는 'BGTile'을 생성

생성된 'BGTile'을 기준으로 게임에서 사용할 'ObjectTile'을 배치

MapEditer – 데이터 입출력 (BinaryData)



TextData폴더

Save & Load

```
void BabaTileMap::Load(string file)
{
    BinaryReader* reader = new BinaryReader(file);
```

```
void BabaTileMap::Save(string file)
{
    BinaryWriter* writer = new BinaryWriter(file);

    writer->UInt(width);
    writer->UInt(height);

    writer->UInt(bgTiles.size());

    for (Object* tile : bgTiles)
    {
        BabaTileMap::Data data;
        data.key = tile->tag;
        data.pos = tile->Position();

        writer->String(data.key);
        writer->Float(data.pos.x);
        writer->Float(data.pos.y);
    }
```

BinaryData를 이용해 Save & Load 구현
생성한 타일의 Key값, 위치등을 저장
이후 모든 Object의 정보를 담고있는 ObjectSample 클래스를 이용해 생성

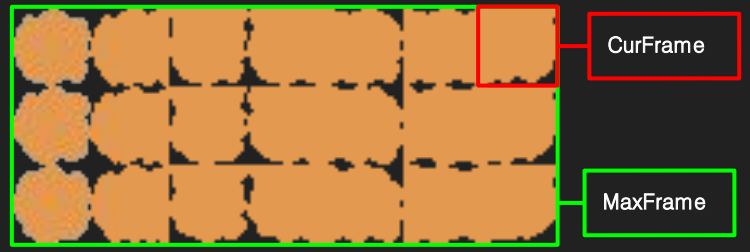
MapEditer / InGame – 최적화 (Instancing)

DrawCall의 최소화를 위해 Instancing을 이용
각 이미지당 한번의 DrawCall을 위한 InstanceQuad를 생성

이미지 Texture데이터, 각 객체의 데이터 InstanceData를 보유

<InstanceData>
위치, 크기, 회전등의 정보를 위한 Transform 행렬
애니메이션을 위한 Frame 데이터

Texture 이미지



이미지를 사용할 Quad와 InstanceBuffer생성

```
class InstanceQuad
{
private:
    const UINT DESC_SIZE

    struct InstanceData
    {
        Matrix transform;
        Vector2 maxFrame;
        Vector2 curFrame;
    };
```

```
InstanceQuad::InstanceQuad(string key, int count)
{
    ObjectSample::InstanceData data = ObjectSample::Get()->Get
    quad = new Quad(data.path, { 0,0 }, { 1 / data.maxFrame.x,
    quad->SetVertexShader(L"Shaders/VertexInstancing.hlsl");
    quad->SetPixelShader(L"Shaders/PixelInstancing.hlsl");

    CreateInstanceBuffer(count);
}
```

InstanceData를 이용해 출력

```
struct VertexInput
{
    float4 pos : POSITION;
    float2 uv : UV;

    matrix transform : INSTANCE_WORLD;
    float2 maxFrame : INSTANCE_MAX;
    float2 curFrame : INSTANCE_CUR;
};
```

```
void InstanceQuad::Render()
{
    instanceBuffer->Set(1);

    quad->SetRender();

    DC->DrawIndexedInstanced(6, objectDatas.size(), 0, 0, 0);
}
```

InGame – 게임기능 (속성)

속성 Object

각 개체마다 변경되는 속성을 부여하기 위해 Action클래스를 상속받은 하위 클래스들을 이용해구현
개체의 속성은 생성되는 하위클래스에 따라 적용
적용된 개체들은 자신의 속성(effect)을 string으로 저장

Action클래스를 설정하는 함수

```
case BabaScene::ActionType::STOP:
    object->effect = "STOP";
    break;
case BabaScene::ActionType::DEFEAT:
    object->effect = "DEFEAT";
    object->SetAction(new Defeat((Transform*)object));
    break;
case BabaScene::ActionType::PUSH:
    object->effect = "PUSH";
    break;
case BabaScene::ActionType::HOT:
    if(object->effect != "PUSH")
        object->effect = "HOT";
    object->SetAction(new Hot((Transform*)object,object->tag));
```

Action클래스

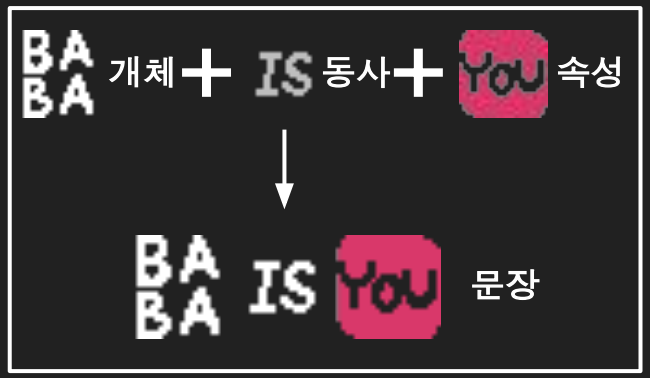
```
class Action
{
public:
    Action();
    virtual ~Action();

    virtual void Update() {};
    virtual void Render();

protected:
    vector<Particle*> particles;
};
```

Action 자식클래스

- Move
- Win
- Defeat
- Sink
- Hot

InGame – 게임기능 (단어조합)

인게임 화면



동사 Object의 위, 아래 또는 좌우로 문장을 완성시킬경우
속성에 해당하는 능력을 개체에 부여

You속성을 가진 개체가 이동이 끝난후 호출
속성에 해당하는 클래스는 외부에 있기에 함수포인터를 이용해 구현

문장 검사 함수

```
void BabaScene::CheckIs()
{
    for (IsObject*& object : propertyIs)
    {
        for (Object*& target : objects)
        {
            if ((target->tag.find("NAME") != string::npos || target->tag.find("PROPERTY
                && target->tag.find("IS") == string::npos)
            {
                if (target->Position() == object->Position() - Vector2(48.0f, 0) && tar
                {
                    if (object->GetHaveObject().left != target)
                        object->RemoveHaveObject();
                    object->GetHaveObject().left = target;
                }
                isMove = false;

                EventManager::Get()->PlayEvent("Check");
            }
```

Move클래스

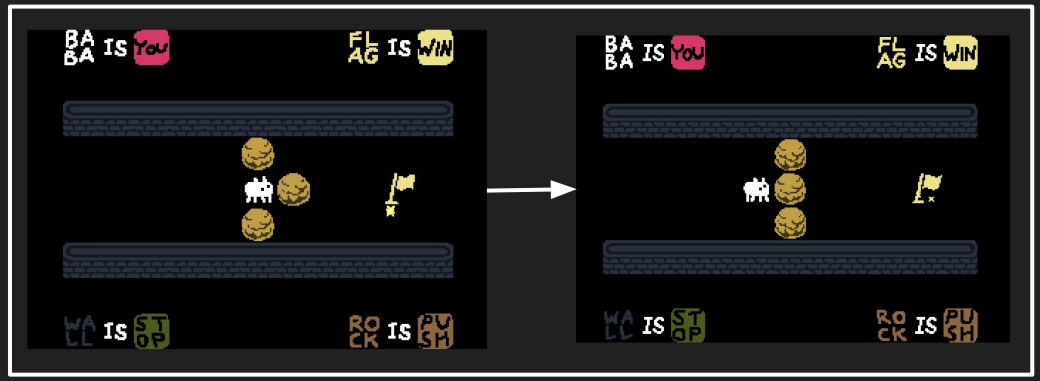
함수포인터 호출

InGame – 게임기능 (되돌리기)

플레이어가 이동을 명령했을때
변경된 타일의 위치, 상태, 애니메이션 등을 Stack에 저장

되돌리기시 Stack에 저장된 데이터를 이용하여 구현

인게임 되돌리기



기존 데이터와 현재데이터 비교함수

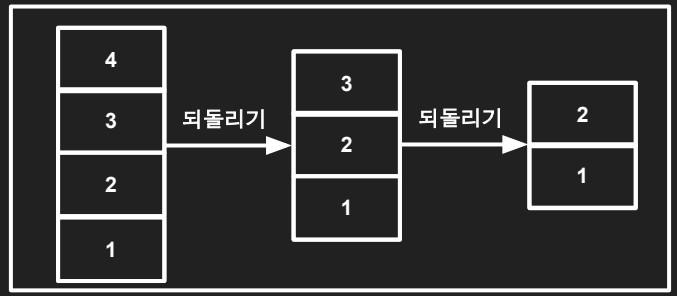
```
void GetBackObject::SetNextData(Object* o
{
    ChangeObjectData tempData;

    for (ChangeObjectData data : prevData
    {
        if (data.object == object)
        {
            tempData.object = object;
            tempData.transform.Position(
            tempData.transform.SetActive(
            tempData.curFrame = tempData.
        }
    }

    datas.push_back(tempData);
```

GetBack함수

```
void GetBackObject::GetBack()
{
    if (saveDatas.size() == 0)
        return;

    vector<ChangeObjectData> tempData;

    tempData = saveDatas.top();

    for (ChangeObjectData data : tempData)
    {
        data.object->Position() += data.transform.Position();
        if (data.object->Active() != data.transform.Active())
            oneMore = true;
        data.object->SetActive(data.transform.Active());
        data.object->GetInstanceData().curFrame = data.curFrame
    }

    saveDatas.pop();
```

기존 데이터를 저장하는 PrevData와
변경후인 현재 데이터를 비교후 saveData에 저장
되돌리기 이후 사용한 정보는 제거

saveDatas



InGame – Particle

파티클 에디터 화면

인게임 적용 화면

게임 이펙트를 위해 파티클 구현
최적화를 위해 Instancing 사용

사용할 이미지, 개수, 지속시간, 색상, 방향, 가속도, 회전, 속도, 생성타이밍, 크기, 지정 변경 등 구현

파티클의 색상, 크기 지정 변경등은 자연스러움을 연출하기위해 선형보간을 이용

맵에디터와 동일하게 Binary데이터로 파티클에 대한 정보를 저장 및 사용

구현된 파티클은 Action클래스에서 호출

Particle실행 함수

```
color.z = LERP(data.startColor.z, data.endColor.z, lifeTime / data.d
color.w = LERP(data.startColor.w, data.endColor.w, lifeTime / data.d

for (UINT i = 0; i < data.count; i++)
{
    if (particleInfos[i].startTime > lifeTime)
    {
        transforms[i].Scale() = { 0, 0 };
        transforms[i].UpdateWorld();
        instances[i].transform = XMMatrixTranspose(transforms[i].Get
        continue;
    }

    particleInfos[i].velocity += particleInfos[i].accelation * DELTA
    transforms[i].Position() += particleInfos[i].velocity
```

개발소감

1. 타일맵 구조의 퍼즐게임의 DrawCall횟수가 너무많아 프레임이 떨어지는 문제가 발생
   Instancing을 이용하여 DrawCall을 줄여 프레임을 향상했습니다.

2. 문장의 완성을 검사하는 함수를 제작할때 함수의 호출 시점을 찾는데에 고민을 많이했습니다.
   플레이어의 이동이 끝이 났을때 검사하는 형태를 채용하여 불필요한 함수호출을 피할수 있었습니다.

3. YOU속성의 개체가 이동할때 이동할 방향에 위치한 PUSH속성 개체를 판별하여 함께 이동해야하는 상황에
   PUSH속성개체를 특정화하는데에 고민했습니다.
   이에 재귀함수를 이용해 플레이어의 이동방향의 PUSH속성 개체를 얻을수있었습니다.

4. 되돌리기 기능을 구현하기 위해 stack을 이용할수있는 좋은 기회였습니다.

이번 포트폴리오를 제작하며 최적화와 코드의구조, STL등을 이용하는 실력이 늘었습니다.