

방방 제작보고서

하운지 이에영 배종성 황규민



목차

최종 결과물 소개

클래스 다이어그램

게임 오브젝트

주요 기능 구현

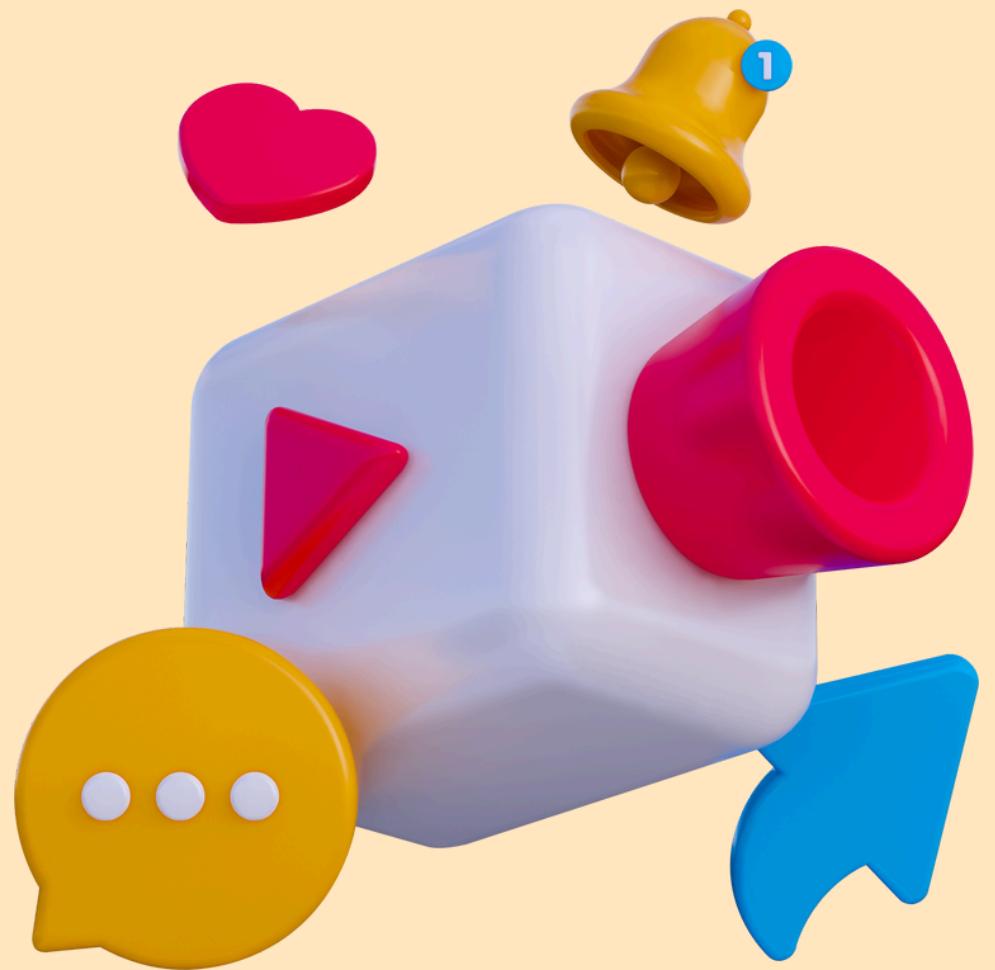
문제 해결 과정

최종 결과물 소개

하이라이트 영상
최종 구현 이미지



| 하이라이트 영상



풀플레이영상 : <https://youtu.be/SdC8TNcZ2sU>



| 최종 구현 이미지

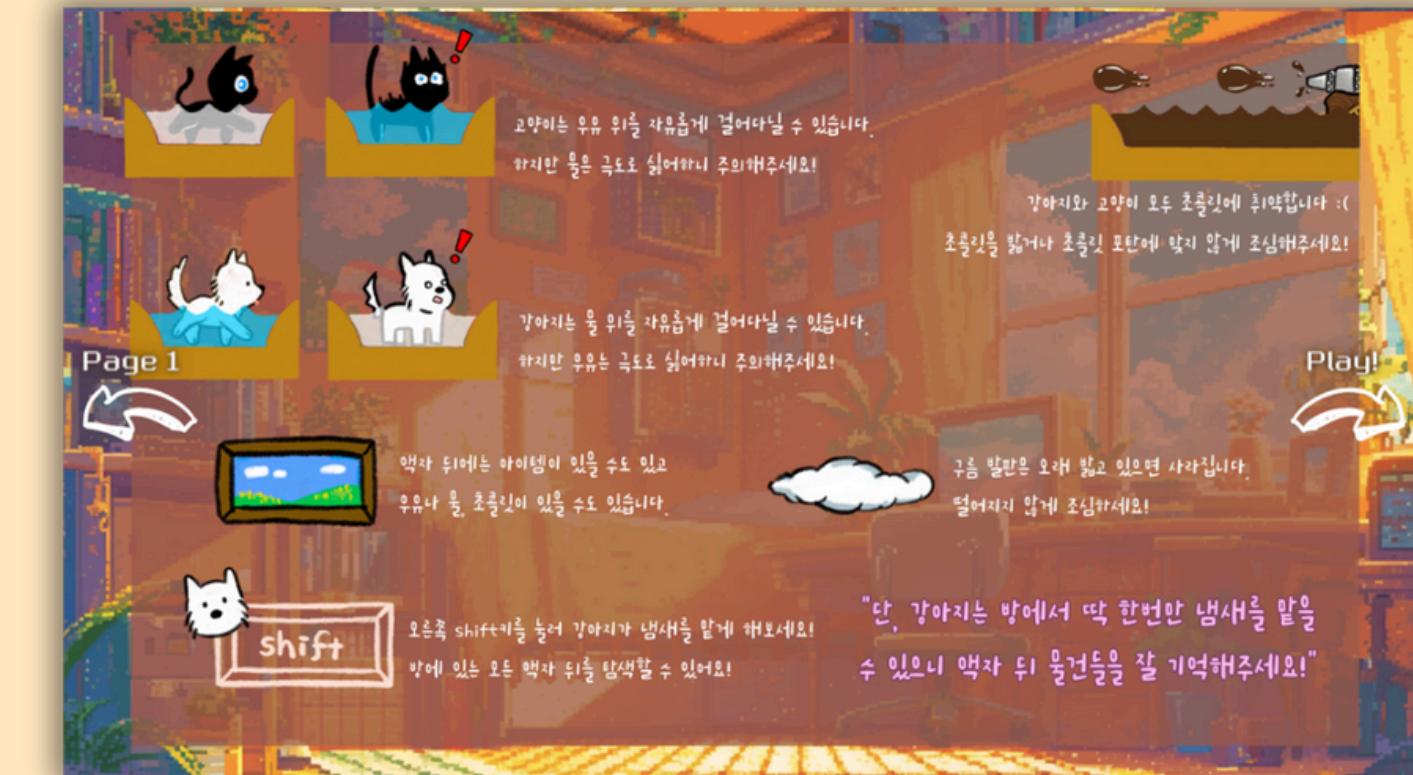
<Intro>



게임 시작 시 가장 먼저 나오는 화면
아무 입력이나 넣으면 메뉴얼 화면으로 이동

| 최종 구현 이미지

<Manual>

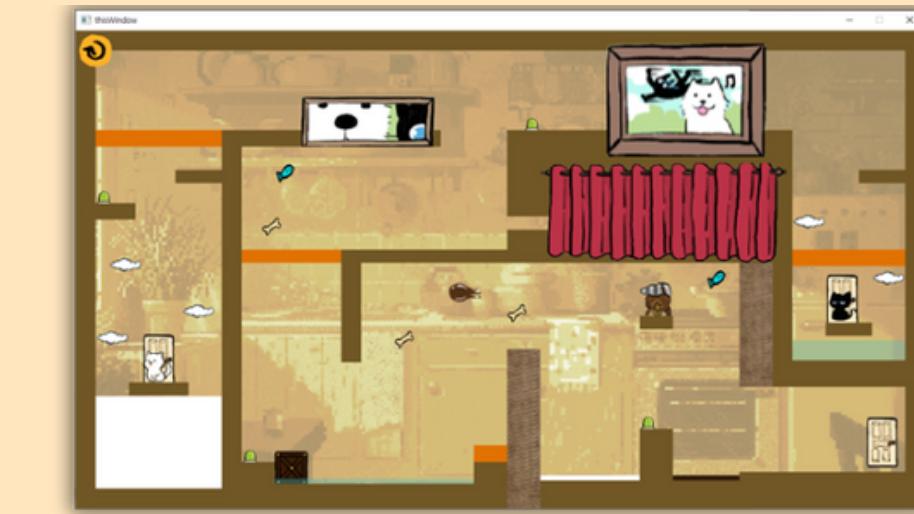
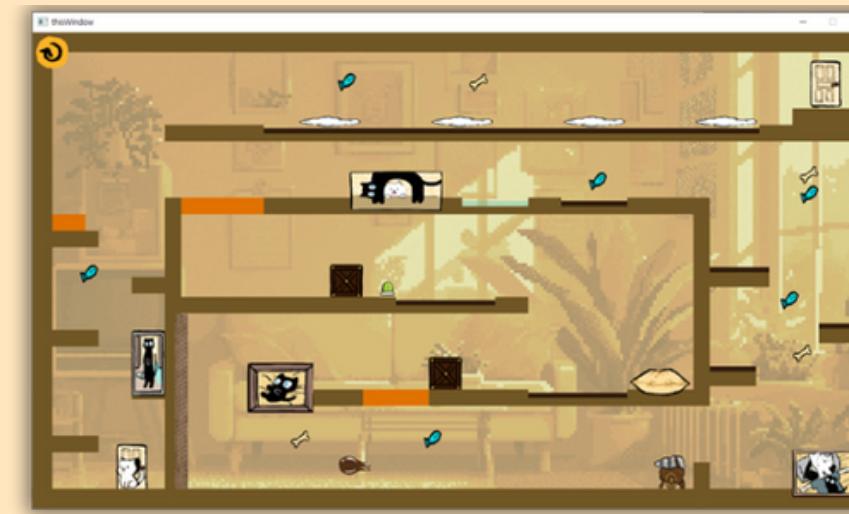
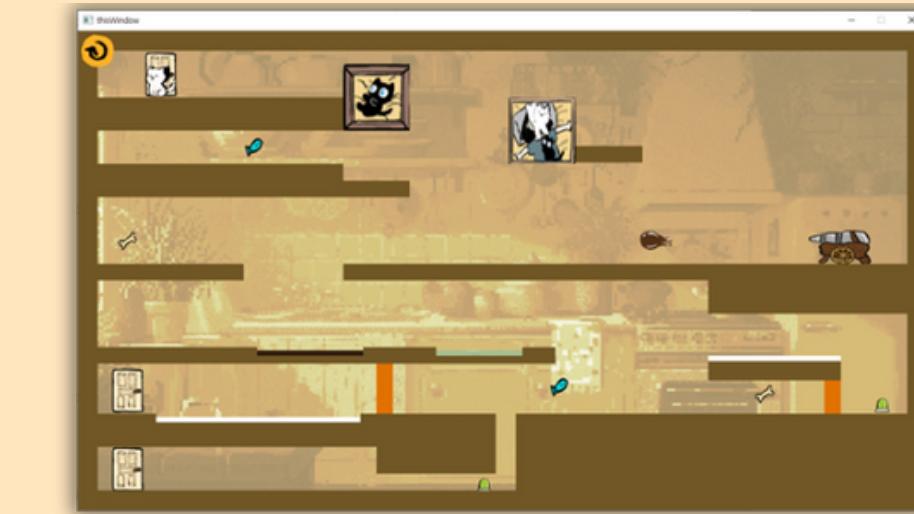


두 장으로 이루어져 있고 왼쪽/오른쪽 화살표를 마우스 좌클릭해 앞장과 뒷장으로 이동 가능
뒷장에서 오른쪽 화살표를 클릭하면 Map1 or 진행 중이던 Map으로 이동



| 최종 구현 이미지

<Map>



난이도 쉬운 순으로 제시
일부 맵에선 Manual에서 설명하지 않은 오브젝트/시스템 존재



| 최종 구현 이미지

-캐릭터 기절 조건-



물, 초콜릿에 접촉
초콜릿 포탄에 접촉



우유, 초콜릿에 접촉
초콜릿 포탄에 접촉

<Game Over>



한 캐릭터라도 기절하면 게임 오버
아무 키나 누르면 Map 재시작

조건 충족 시 놀라는 이미지로 약 2초 지속 후 게임 모버 화면 이동
놀라는 이미지가 지속되는 동안 해당 캐릭터만 움직이지 않음

| 최종 구현 이미지

<Game Clear>



게임 클리어 조건 달성 시 나오는 화면
획득한 뼈다귀, 생선 갯수 표시
Map 재시작하거나 다음 Map으로 이동



| 최종 구현 이미지

<ESC>



Map에서 ESC를 누르면 나오는 화면

마우스 좌클릭으로 선택

이어하기 - Map으로 돌아가기
처음으로 - Intro로 이동, 전체 초기화
가이드 - Manual로 이동
나가기 - 게임 종료(EXIT)

| 최종 구현 이미지_스토리



<엔딩 컷씬>



게임 엔딩 컷씬으로 스토리 유추 가능

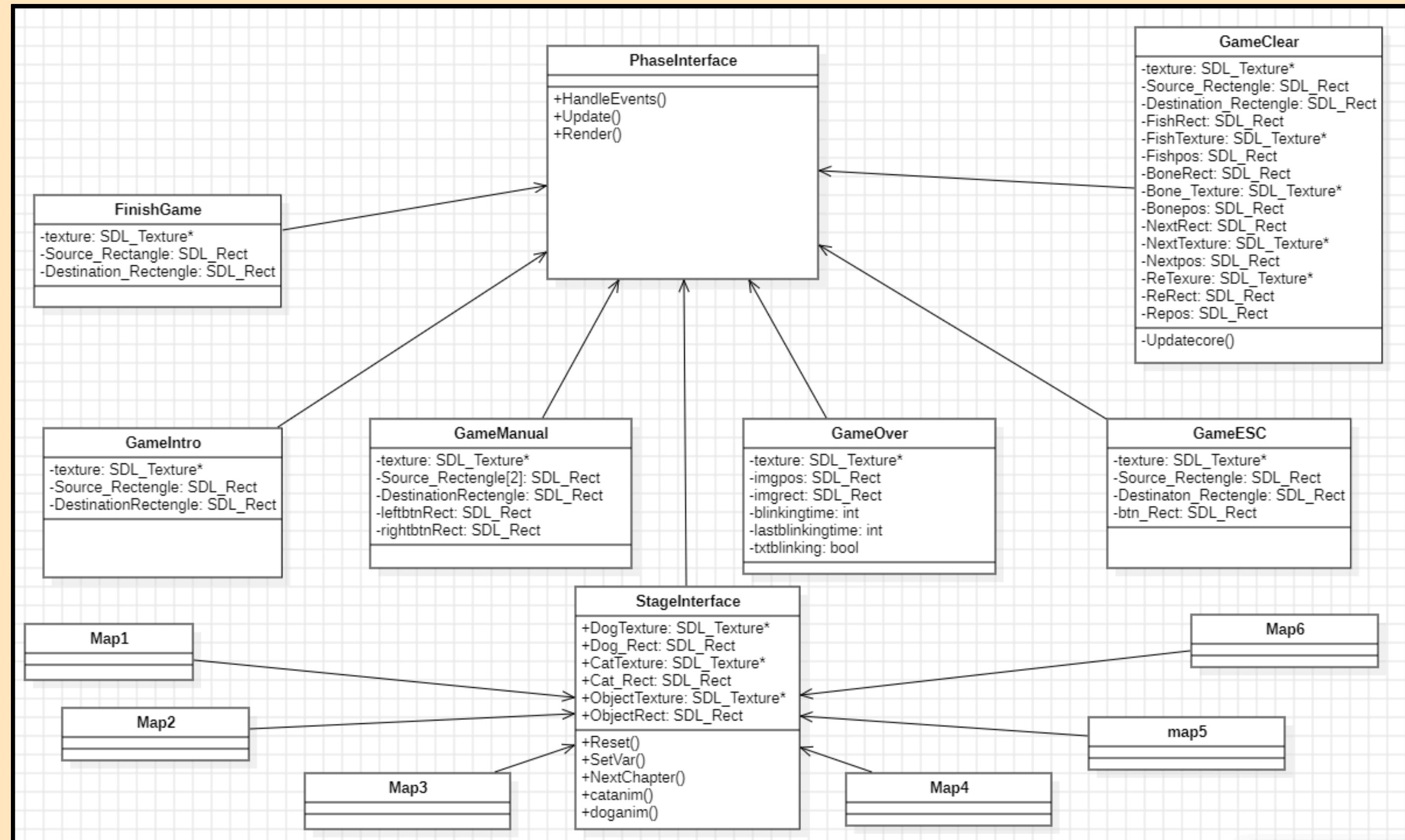
고양이와 강아지가 심심하다며 집안을 돌아다니며 나가는 문을 찾았던 이유는
밖으로 나가 주인을 마중 나가기 위해서였다.
마지막에 주인이 오고 셋은 기뻐한다.

클래스 다이어그램

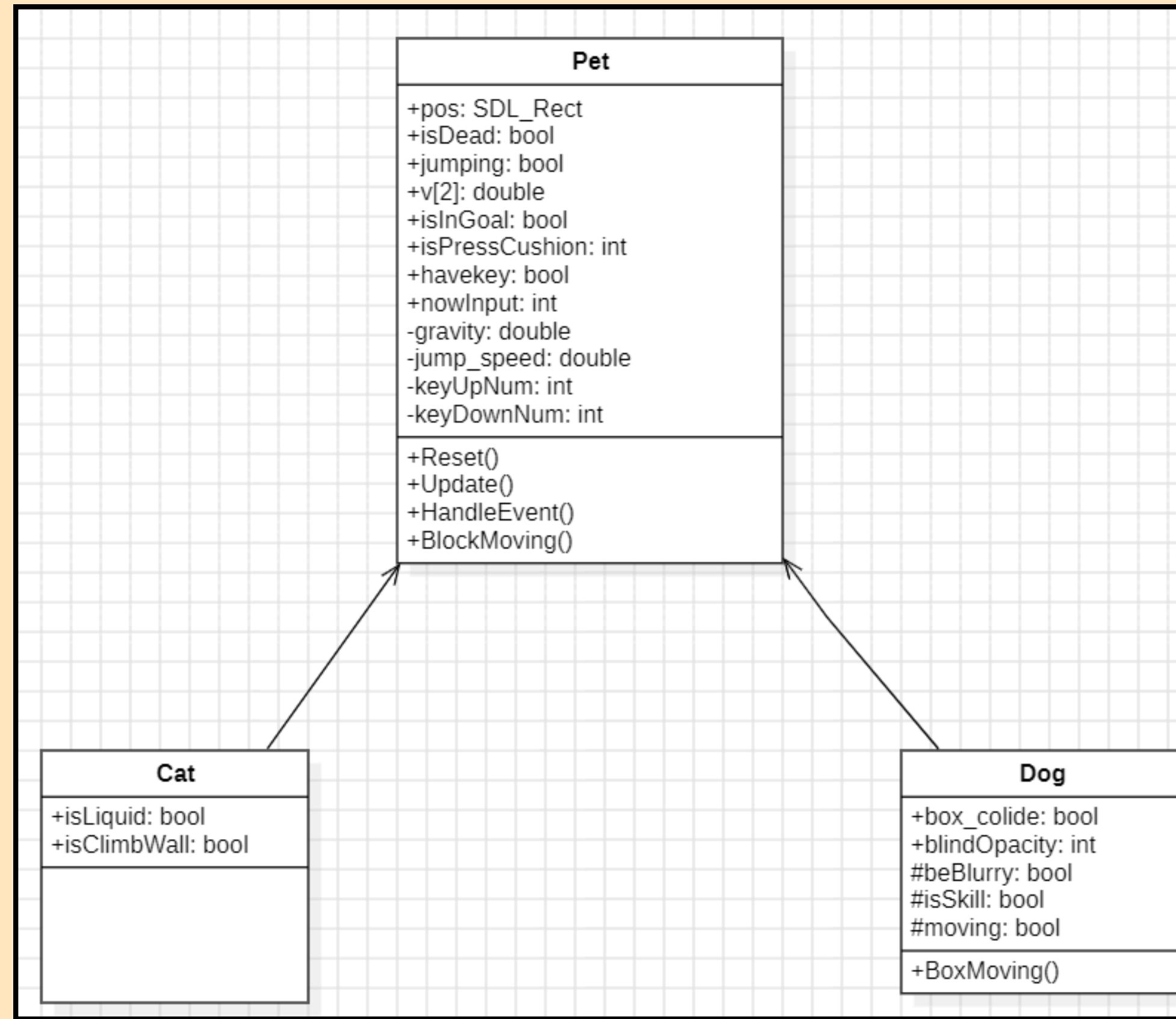
페이지 다이어그램
Pet 다이어그램



| 페이즈 디자인그램



| Pet 디아그램



게임 오브젝트

게임 오버 관련 오브젝트 Cat 상호작용 오브젝트

Dog 상호작용 오브젝트 버튼과 움직이는 발판

기타 오브젝트



| 게임 오버 관련 오브젝트 Liquid



강아지는 우유, 고양이는 물에 닿으면 기절 상태가 되고 게임 오버
초콜릿에 닿으면 두 캐릭터 모두 기절 상태



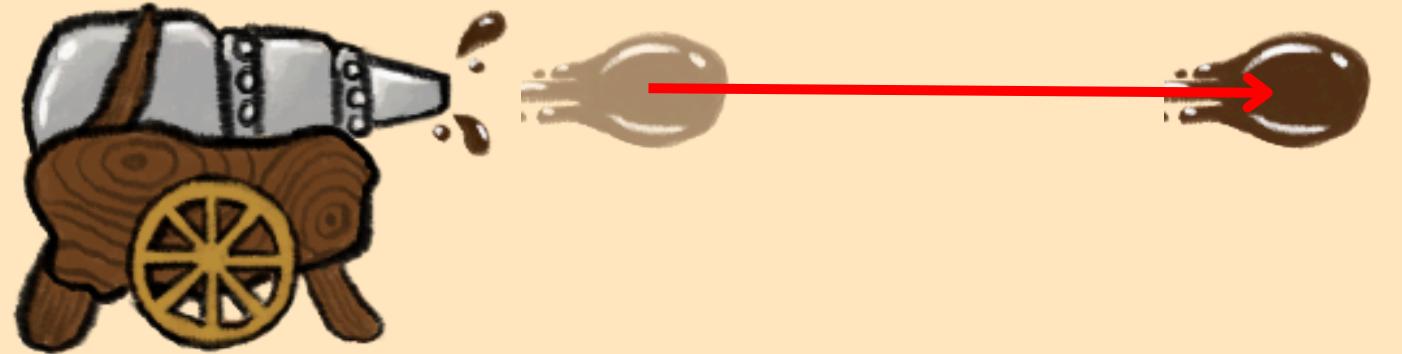
| 게임 오버 관련 오브젝트 SwellingLiquid



Map6에 등장하는 액체 오브젝트
시간이 지남에 따라 점차 차오름



| 게임 오버 관련 오브젝트 Cannon+Missile

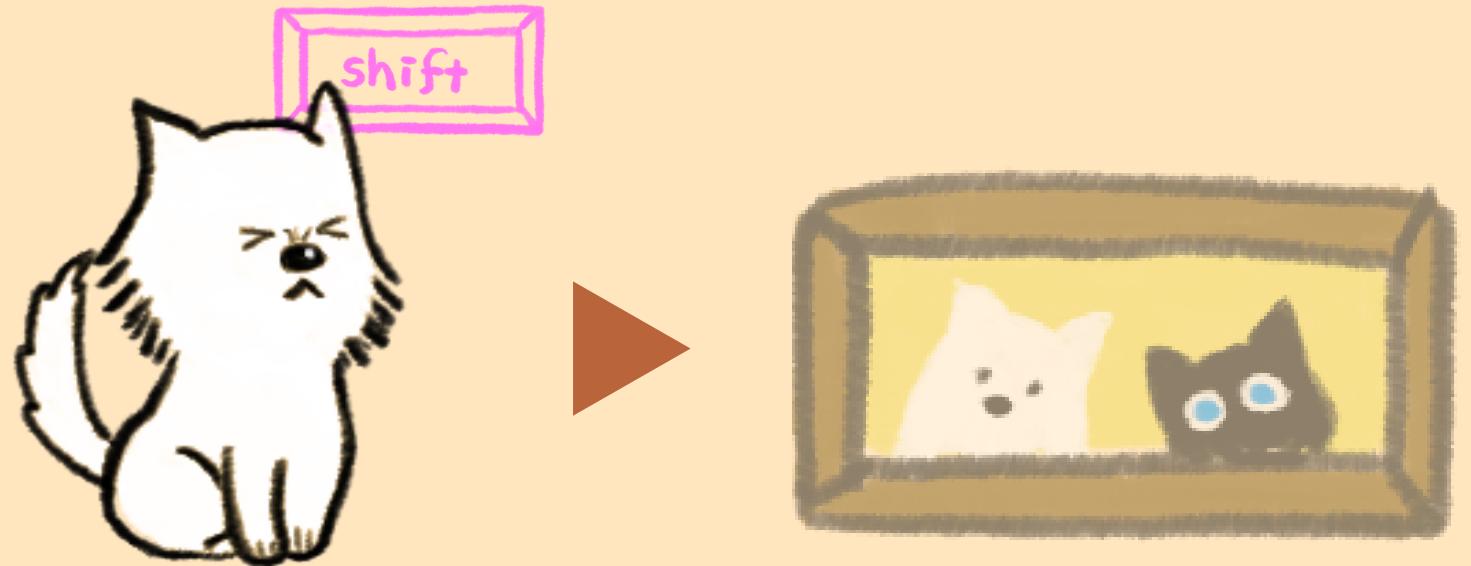


- Missile은 초콜릿 포탄을 의미
- Cannon의 앞쪽에서 Missile이 일직선으로 발사됨
- 발사 방향은 왼쪽 혹은 오른쪽으로 초기 설정 가능



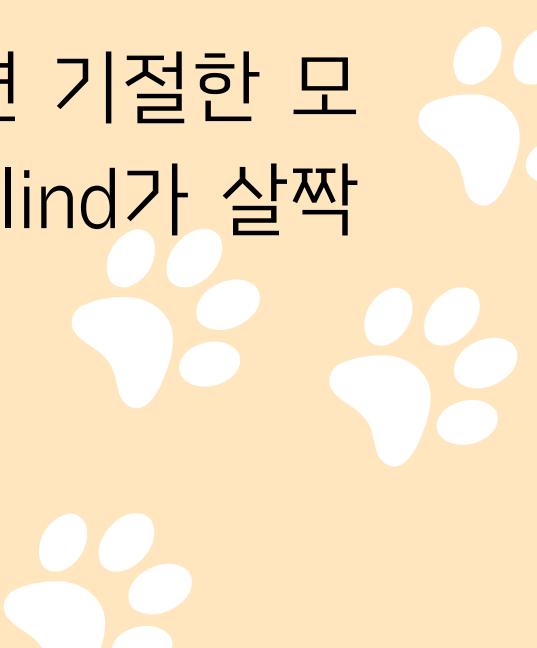
- Missile은 충돌 여부 감지 후 다른 오브젝트들과는 다르게 Pet의 중점과 Missile의 중점 사이의 거리가 일정 값 이내일 경우에 닿은 것으로 판정
=좀 더 널널한 판정

| Dog 상호작용 오브젝트 Blind



- 오른쪽 shift를 한 번 누르면 강아지가 sniff skill을 사용, 발동 시간 동안 움직일 수 없음
- 그 동안 액자 혹은 커튼으로 그려진 blind 오브젝트 가 희미해지며 뒤에 있는 물체 확인 가능
- 스킬은 한 맵에 한번만 사용 가능

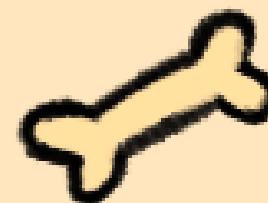
- Blind뒤에서 강아지나 고양이가 기절하면 기절한 모습을 보여주는 동시에 pet이 달아있는 blind가 살짝 투명해짐



| Dog 상호작용 오브젝트 Box & Bone



- 강아지만 밀 수 있는 박스
- 고양이는 박스 위에 올라탈 수 있으나, 올라탄 상태에서 강아지가 박스를 밀면 박스와 함께 이동하지 않음
-> 게임의 난이도와 재미를 높이기 위함



- 강아지만 먹을 수 있는 간식
- 맵마다 전체 갯수가 정해져 있고 강아지가 먹을 시 점수를 획득

| Cat 상호작용 오브젝트 Liquid Wall

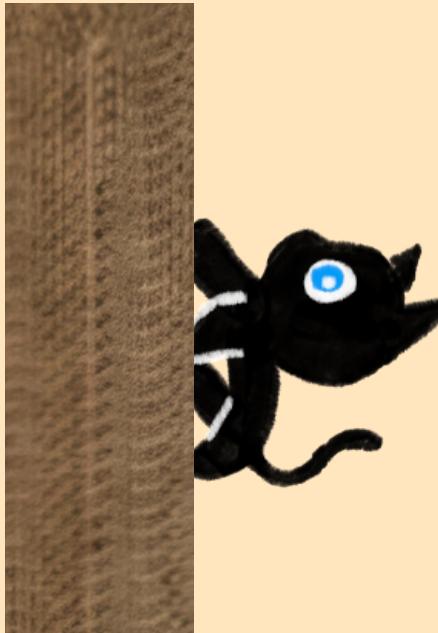


강아지는 이동하지 못하게 하여 구현
고양이가 들어가면 액체처럼 우글거리는 애니메이션이 재생되며 이동 가능

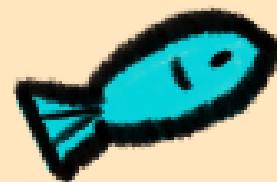


| Cat 상호작용 오브젝트

Climb Wall & Fish



- 고양이만 오를 수 있는 수직 벽
- 고양이가 climb wall에 닿으면 중력이 반대 위쪽으로 작용
- 일반 중력처럼 여기에도 제한이 존재
- $V[1]=-1.5f;$

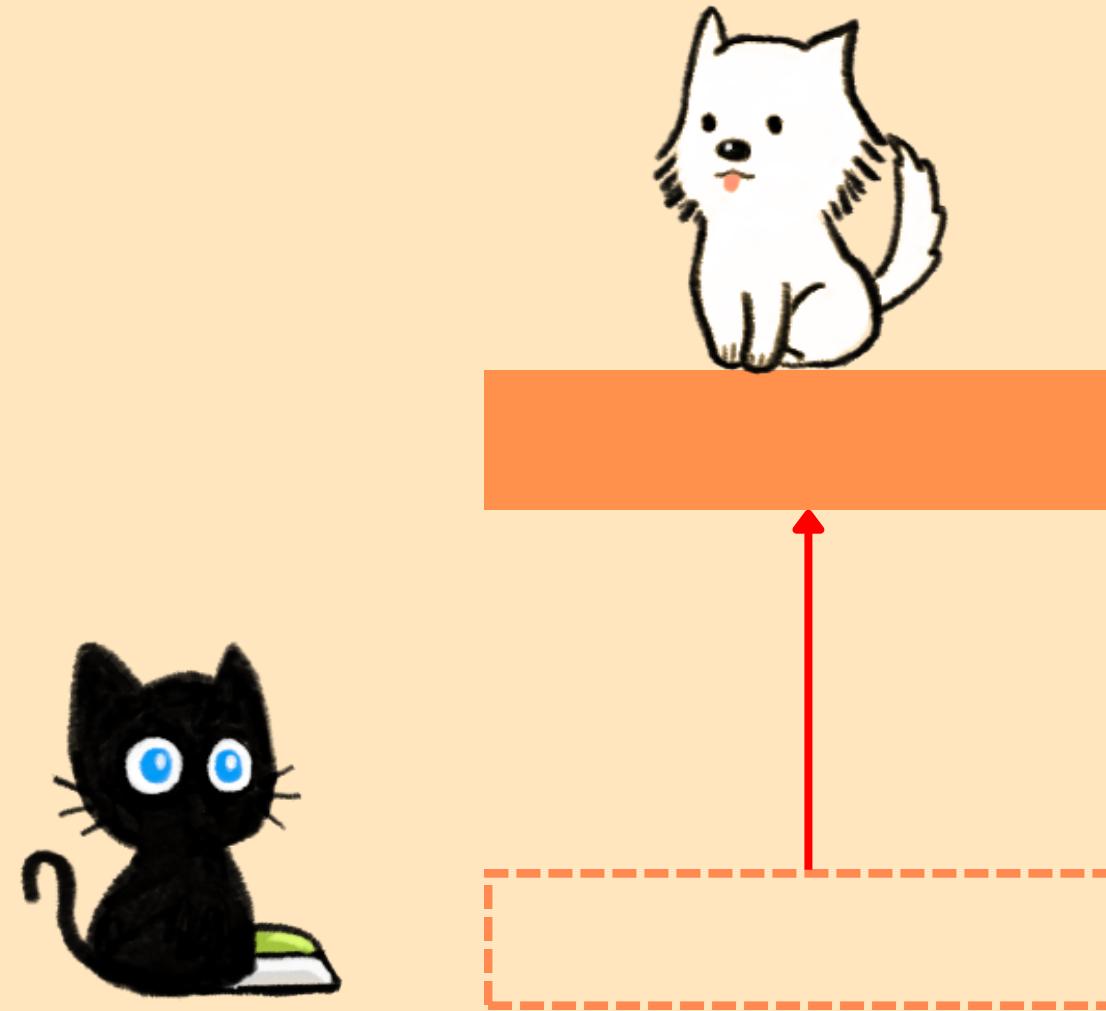


- 고양이만 먹을 수 있는 간식
- 맵마다 전체 갯수가 정해져 있고 고양이가 먹을 시 점수를 획득



| 버튼과 움직이는 발판

Button + Scaffold



- 버튼을 누르면 발판이 출발 위치에서 목표 위치까지 이동
- 버튼에서 내려오면 발판은 다시 출발 위치로 이동
- 버튼 여러 개로 한 개의 발판을 움직이거나, 버튼 하나로 여러 개의 발판을 움직일 수 있다.
- 이동하려는 방향 아래에 강아지나 고양이가 있다면 이동 멈춘다. (단, 아래에서 위로 올라가는 발판을 Pet이 막고 있을 때 멈추는 코드는 구현되어 있지 않다)

| 기타 오브젝트

FadeFloor



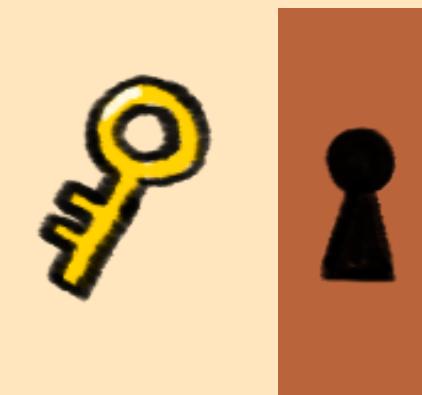
한 번 밟으면 희미해 지다
완전히 사라짐

Cushion



밟으면 더 높게 점프 가능

Key(Lock)



열쇠를 얻으면 열쇠를 얻은 Pet은
Lock에 닿았을 때 잠금을 없앨 수
있음. 그 전에는 잠금된 곳을 지나
갈 수 없음

주요 기능 구현

Pet 움직임 조작

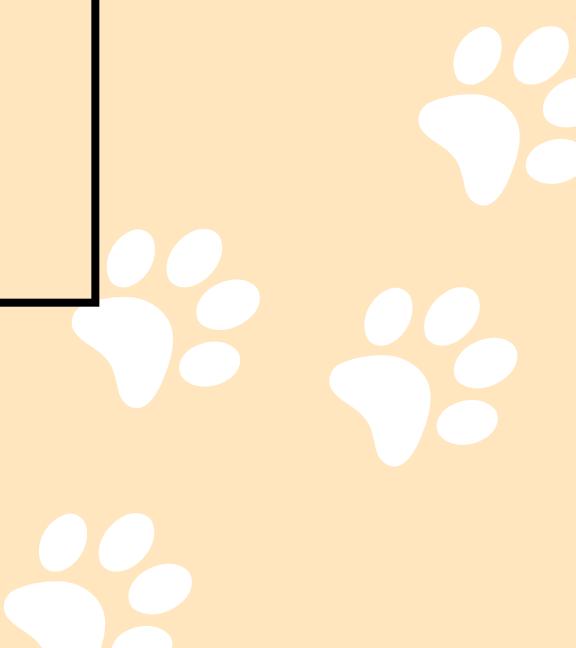
충돌 감지

Block Moving



| Pet 움직임 조작

좌우 이동	<p>std::vector<int> inputs에 입력순서대로 저장 누르던 키를 때면 해당키는 inputs에서 삭제 가장 최근에 누른 키 (가장 최근에 inputs에 저장된 값)를 불러와 작동을 지시 ->List로 구현했으면 더 수월했을 것 시간 문제 상 고치지 못했음</p>
점프	<p>좌우 이동과 개별적으로 작동 Jump Speed만큼 y값을 줄이지만 중력이 계속 더해지기 때문에 하락한다.</p>
중력	<p>Cat, dog, box에 중력이 계속 더해진다.</p>
중력 제한	<p>제한이 없다면 움직임이 더 자연스럽지만 충돌 판단 문제로 최대 6.5로 제한</p>



| Pet 움직임 조작

```
void Cat::HandleEvent(SDL_Event event)
{
    Pet::HandleEvent(event);

    switch (event.type)
    {
        case SDL_KEYDOWN:

            if (event.key.keysym.sym == SDLK_a)
            {
                v[0] = -1.0f; keyDownNum = 1;
            }
            else if (event.key.keysym.sym == SDLK_d)
            {
                v[0] = 1.0f; keyDownNum = 2;
            }
            else if (event.key.keysym.sym == SDLK_w)
            {
                if (jumping == false)
                {
                    v[1] = jumpSpeed();
                    Mix_PlayChannel(2, meow, 0);
                    jumping = true;
                }
            }
    }
}
```

1

이동키 입력을 KeyDownNum 변수의 정수값으로 저장
Inputs 벡터의 마지막 원소로 저장

```
if (keyDownNum != 0)
{
    for (int i = inputs.size() - 1; i >= 0; i--)
    {
        if (inputs[i] != keyDownNum && keyDownNum != 0)
        {
            if (i == 0) { inputs.push_back(keyDownNum); }
            else { break; }
        }
        keyDownNum = 0;
    }
    break;
}
```

2

| Pet 움직임 조작

```
case SDL_KEYUP:  
    if (event.key.keysym.sym == SDLK_a)  
    {  
        keyUpNum = 1;  
    }  
    else if (event.key.keysym.sym == SDLK_d)  
    {  
        keyUpNum = 2;  
    }  
    else if (event.key.keysym.sym == SDLK_w)  
    {  
        //catKeyUp = 3;  
    }
```

3

```
if (keyUpNum != 0)  
{  
    for (int i = inputs.size() - 1; i >= 0; i--)  
    {  
        if (inputs[i] == keyUpNum)  
        {  
            //dog->v[0] = 0.0f;  
            inputs.erase(inputs.begin() + i);  
            keyUpNum = 0;  
            break;  
        }  
    }  
}
```

4

누르던 키를 떼면 그 값을 KeyUpNum의 정수값으로 저장
Inputs 벡터 내에서 KeyUpNum과 같은 원소 삭제

| 충돌 감지

1) `SDL_HasIntersection()` 으로 충돌 여부 탐지

Ex) 초콜릿, 우유, 물 닿았을 때 사망 / climb wall에 닿고 올라가는 것 등..

`SDL_HasIntersection()`

인수로 들어오는 rect 영역이 겹치면 `SDL_TRUE`,
겹치지 않으면 `SDL_FALSE` 반환

2) 직접 코드로 탐지

벽과의 거리가 n이하일 때

```
void Dog::Update(double timestep_s)
{
    if (v[1] <= 6.5) { v[1] += gravity; }

    Pet::Update(timestep_s);

    //MILK///////////
    for (Liquid l : liquid)
    {
        if (SDL_HasIntersection(&l.liquidPos, &pos))
        {
            if(nowInput == 1 || nowInput == 2){Mix_PlayChannel(-1, liquidSound, 0); }

            if (l.liquidClass == "milk")
            {
                isDead = true;
            }
        }
    }
}
```

| Block Moving



| Block Moving

```
void Pet::BlockMoving(SDL_Rect obst)
{
    if (pos.y + pos.h >= obst.y + 7 &&
        pos.y <= obst.y + obst.h - 7)
    {

        if (pos.x < obst.x + obst.w / 2)
        {
            //std::cout << "left\n";
            //pos.x = min(pos.x, obst.x - pos.w);
            pos.x = obst.x - pos.w;
            //jumping = true;
        }
        else if (pos.x + pos.w > obst.x)
        {
            //std::cout << "right\n";
            //pos.x = max(pos.x, obst.x + obst.w);
            pos.x = obst.x + obst.w;
            //jumping = true;
        }
    }
}
```

벽의 위아래에 있는지 판단하는 것은 7의 여분을 둠
->너무 여분을 적게 하면 점프가 어려워짐
->가속도가 너무 커지면 뚫고 갈 수 있기 때문에 여분 필요

| Block Moving

```
else
{
    if (pos.y + pos.h <= obst.y + obst.h / 2)
    {
        //std::cout << "up\n";

        //pos.y = obst.y - pos.h;
        pos.y = obst.y - pos.h < pos.y ? obst.y - pos.h : pos.y;
        //pos.y = std::min(obst.y - pos.h, pos.y);
        v[1] = 0;
        jumping = false;
    }
    else
    {
        //std::cout << "down\n";

        //pos.y = obst.y + obst.h;
        pos.y = obst.y + obst.h > pos.y ? obst.y + obst.h : pos.y;
        //pos.y = std::max(obst.y + obst.h, pos.y);
        v[1] = 0;
        jumping = true;
    }
}
```

Pet이 벽의 아래쪽에 부딪히면 즉시 아래로 하강
Pet이 벽의 위를 밟으면 다시 점프를 할 수 있다

문제 해결 과정

게임 기반에 대한 문제

게임 물리에 대한 문제

게임 아트에 대한 문제

게임 애니메이션에 대한 문제



| 게임 기반에 대한 문제



? 어떻게 버튼이 눌러졌을 때 원하는 방향과 거리로 발판을 이동시키지?

? 어떻게 플레이어와 벽이 만난 걸 알아내지?



💡 이동 방향 벡터를 구하고 이동 거리 분의 1씩 해당 방향으로 이동 시켜 사방으로의 움직임이 가능한 함수 사용

💡 `SDL_HasIntersection`으로 벽에 닿았는지 확인하고, 케이스 분류를 나눠서 벽이 위, 아래, 좌, 우 어디에 있는지 확인



| 게임 물리에 대한 문제



? 등가속도에서 이동거리를 어떻게 적분 없이 나타내지?

? 어떻게 뉴턴 2법칙을 쓰지 않고 충돌을 나타내지?



💡 직접 적분을 구현하지 않고 리만합으로 유사 적분을 구현

💡 오브젝트와 일정거리만큼 가까워졌으면 계(system)를 이뤄서 같이 이동시킴



| 게임 아트에 대한 문제



? 생성형 AI*로 의도하고자 한 이미지가 나오지 않았을 때 어떻게 이미지를 생성하거나 수정하지?



- 💡 기준이 되길 원하는 이미지를 찾고, /describe에 넣고 핵심 단어들을 추출하고 해당 단어에 맞는 이미지를 생성
- 💡 디테일한 부분은 영역을 지정해 다시 프롬프트를 작성하거나 포토샵으로 수정

| 게임 애니메이션에 대한 문제



? 움직임에 맞는 스프라이트를 어떻게 처리하지?

? 스프라이트의 반전은 어떻게 적용하
지?



💡 다음 움직임인 rect를 동적 배열에 넣고 기존의 rect는 지워서 구현

💡 Keyinput을 통해 왼쪽과 오른쪽을 구분하고 기존의 움직임에서 방향에 따른 경우를 추가해서 경우를 늘림



Thank You!