WinApi를 활용한 C++ 콘솔 프로젝트

BaBa Is You 모작

목차

BaBa Is You 소개

- 개발 : Arvi Teikari (Hempuli)
- 출시 년도 : 2019
- 장르 : 퍼즐
- 주요 수상 : IGF 2018 수상

BaBa Is You 규칙 소개

속성 부여

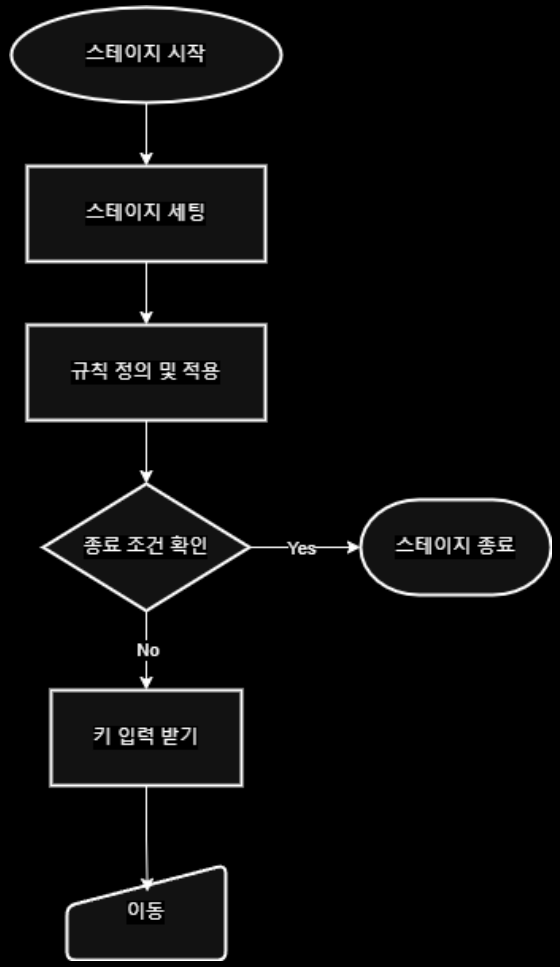
Noun Is Property

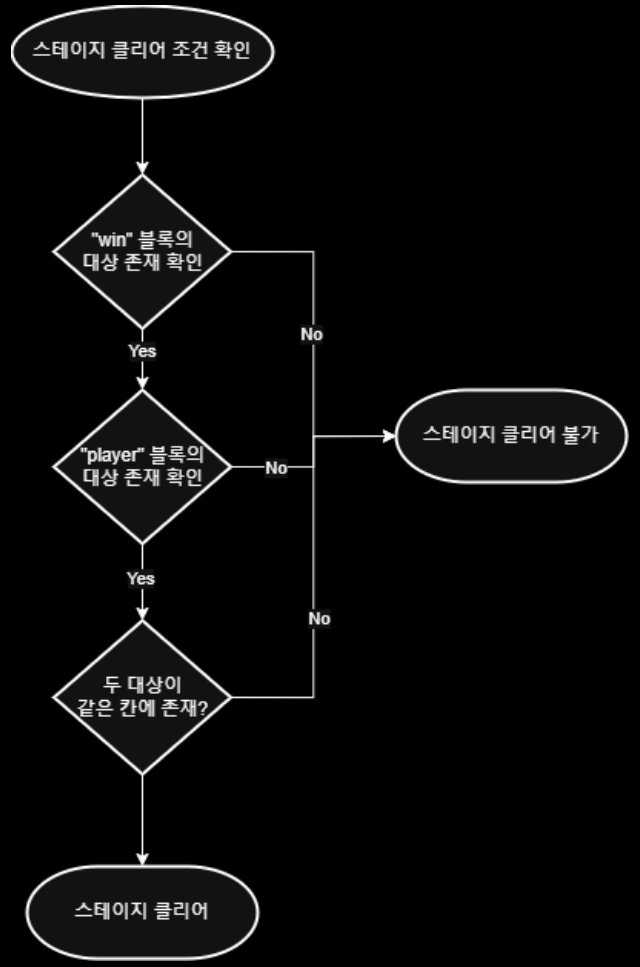
물체 변경

Noun Is Noun

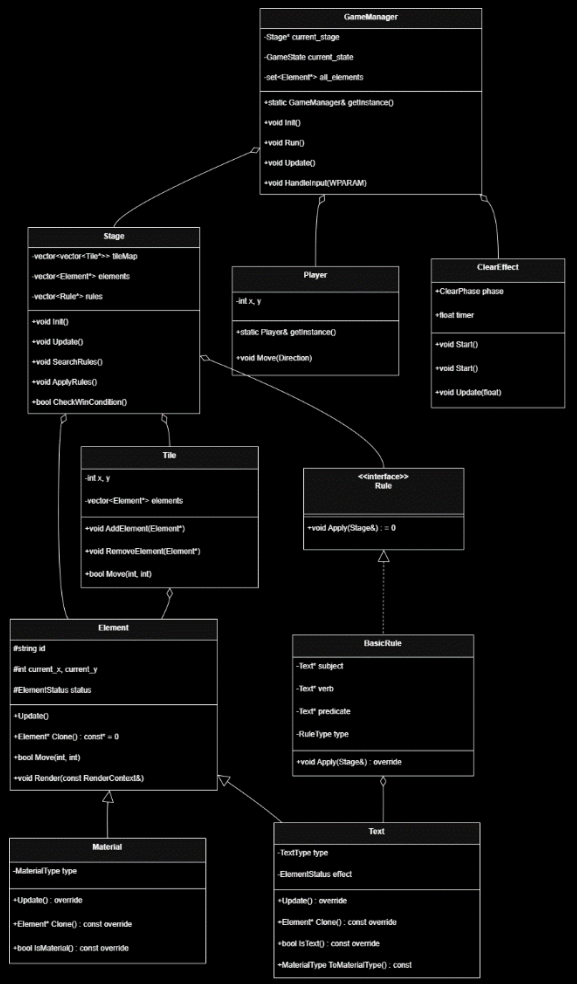
모작 시연 영상

플로우차트


스테이지 시작
스테이지 세팅
규칙 정의 및 적용
종료 조건 확인
Yes
스테이지 종료
No
키 입력 받기
이동


스테이지 클리어 조건 확인
"win" 블록의 대상 존재 확인
No
Yes
"player" 블록의 대상 존재 확인
No
스테이지 클리어 불가
Yes
두 대상이 같은 칸에 존재?
No
스테이지 클리어

클래스 다이어그램

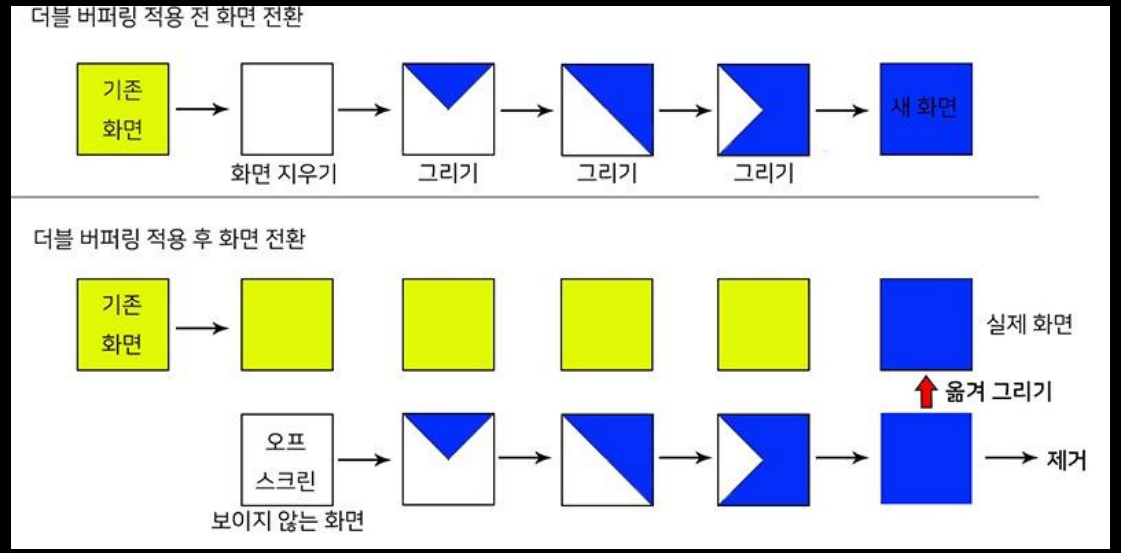


- GameManager에서 Stage, Player, ClearEffect 참조
- Stage에서 Element, Tile, Rule 참조
- Tile에서 Element 참조
- BasicRule에서 Text 참조
- Element는 부모 클래스이고, Text, Material 자식 클래스 존재
- Rule은 인터페이스로 BasicRule에서 실제 구현이 이루어져 있음
- GameManager, Player는 싱글톤으로 구현

더블 버퍼링

```cpp
void GameManager::Render(HDC hdc, int m, int n) {
    // 1. 화면 크기 가져오기
    RECT rect;
    GetClientRect(hWnd, &rect);
    int screenWidth = rect.right - rect.left;
    int screenHeight = rect.bottom - rect.top;

    // 2. tileSize 계산
    if (current_stage != nullptr)
    {
        m = current_stage->GetCols();
        n = current_stage->GetRows();
    }
    int tileW = (screenWidth - 2 * MARGIN_WIDTH) / m;
    int tileH = (screenHeight - 2 * MARGIN_HEIGHT) / n;
    int tileSize = min(min(tileW, tileH), MAX_TILESIZE);

    // 3. 중앙 정렬용 오프셋
    int offsetX = (screenWidth - (tileSize * (m + 2))) / 2;
    int offsetY = (screenHeight - (tileSize * (n + 2))) / 2;

    // 4. 더블 버퍼링 준비
    HDC memDC = CreateCompatibleDC(hdc);
    HBITMAP memBitmap = CreateCompatibleBitmap(hdc, screenWidth, screenHeight);
    HBITMAP oldBitmap = (HBITMAP)SelectObject(memDC, memBitmap);
    Graphics g(memDC);
    g.Clear(Color(50, 50, 50)); // 배경 검정

    // 5. 렌더링 컨텍스트 생성
    RenderContext ctx = { memDC, &g, tileSize, offsetX, offsetY };

    // 6. 현재 스테이지 렌더링
    if (current_stage)
        current_stage->Render(ctx);
    ClearRender(g, screenWidth, screenHeight);
    //player.Render(ctx);

    // 7. 화면에 복사
    BitBlt(hdc, 0, 0, screenWidth, screenHeight, memDC, 0, 0, SRCCOPY);
    SelectObject(memDC, oldBitmap);
    DeleteObject(memBitmap);
    DeleteDC(memDC);
}
```

하나의 버퍼에 그림을 그리고,
완료되면 다른 버퍼로 전환해 깜빡임 없이 출력하는 기술



규칙 정의

- 플레이어의 입력마다 기존 규칙을 초기화하고 새 규칙을 정의
- Text 클래스에서 verb 객체를 탐색
- verb 기준 좌우 또는 위아래 방향으로 text가 있는지 확인
- verb 앞 단어가 Noun인지 검사
- verb 뒤 단어가 Noun이면 '형태 변환 규칙', Property면 '속성 부여 규칙'으로 판단
- 생성된 Rule 객체를 Stage의 Rule List에 추가

규칙 적용

- **형태 변환 규칙**: 첫 번째 명사의 오브젝트를 두 번째 명사의 오브젝트로 교체
- **속성 부여 규칙**: 주어에 해당하는 명사 오브젝트에 속성 부여
  - Update()에서 속성에 따른 동작 처리

물체 이동 알고리즘

- You 속성을 가진 모든 Material이 이동 시도
- 이동 방향 타일에 Stop이 있으면 즉시 이동 중단
- 타일에 Push 오브젝트가 있으면 다음 타일로 재귀 이동 시도
- 다음 타일이 비어 있거나 Push가 없으면 재귀 종료 (true 리턴)
- 밀 수 없는 상황이면 재귀 종료 (false 리턴)
- 밀 수 있으면 Push 오브젝트부터 순서대로 이동
- 재귀 호출이 true면 현재 오브젝트도 이동
- 최종적으로 이동이 불가능하면 아무것도 이동하지 않음

사용한 기법

디자인 패턴

- 싱글톤 : GameManger, Player

STL

- All element : 중복 없는 set
- All stage : vector로 순서 고려
- NameToType : unordered_map
- TileMap : 이중 vector로 좌표 구현

OOP 4원칙

캡슐화

```
class Element
{
protected:
    string id = "None";
    int current_x = 0, current_y = 0;
    float draw_x = 0.0f, draw_y = 0.0f;
    int target_x = 0, target_y = 0;
    float speed = 18.0f;
    wstring image_path = L"";
    Image* image = nullptr;
    Color color = Color(0, 0, 0, 0);
    vector<ElementStatus> status = { ElementStatus::Push };
    ElementStatus temp_status = ElementStatus::Push;
    static Image* glow_effect;
    static set<Element*> all_elements;

public:
    Element(string _id, wstring _image_path, Color _color, ElementStatus _status = ElementStatus::None);
    Element(const Element& other);
    virtual ~Element();
    virtual void Update();
```

추상화

Element.h

```
virtual bool IsText() const { return false; }
virtual bool IsMaterial() const { return false; }
```

Material.h

```
virtual bool IsMaterial() const override { return true; }
```

Text.h

```
virtual bool IsText() const override { return true; }
```

상속

```
class Text : public Element
{
private:
    TextType type;
    ElementStatus effect;
    static const unordered_map<string, MaterialType> nameToType;

public:
    Text(string _id, wstring _image_path, Color _color, TextType _type, ElementStatus _eff
    virtual void Update() override;
    virtual void Reset() override { Element::Reset(); status = { ElementStatus::Push }; }
    virtual Element* Clone() const override { return new Text(*this); }
    virtual bool IsText() const override { return true; }
    TextType GetTextType() const { return type; }
```

다형성

```
Element* CopyFromCatalog(MaterialType type);
Element* CopyFromCatalog(const string& id);
```

향후 개선사항

코드 최적화

클래스 정리

메모리 관리

비효율적인 코드 개선

추가 스테이지

새로운 규칙

새로운 키워드

감사합니다