PORTFOLIO

김동현

E-mail : clxlzl1234@gmail.com

헬테이커

개인 포트폴리오

개발 인원 : 1인

개발 기간 : 7일

제작 언어 : C++

제작 툴 : Visual Studio

리버 시티 걸즈

팀 포트폴리오

개발 인원 : 5인

개발 기간 : 7일

제작 언어 : C++

제작 툴 : Visual Studio

협업 : GitHub

마녀의 집

팀 포트폴리오

개발 인원 : 5인

개발 기간 : 14일

제작 언어 : C++

제작 툴 : Visual Studio

협업 : GitHub

케이디의 Cubic Music

개인 포트폴리오

개발 인원 : 1인

개발 기간 : 6일

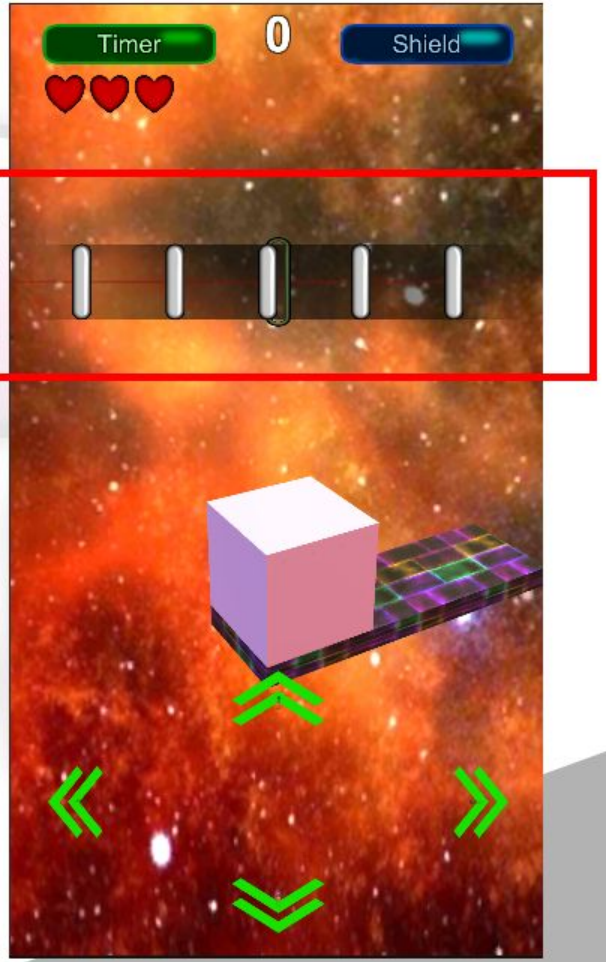
제작 언어 : C#

제작 엔진 : Unity

Cubic Music

영상링크 :
GitHub :https://github.com/mmmdong/CubicMusic

1. 오브젝트 풀링



사용될 노트를 오브젝트 풀링을 이용하여 가비지 콜렉터가 메모리를 처리되는 방식이 아닌 배열을 이용하여 메모리를 할당 및 해제 할 수 있도록 처리했습니다.

노트를 담을 ObjectInfo 클래스를 생성하여 프리팹으로 만들어진 노트를 담았고, 배열은 선입선출이 가능한 Queue를 사용했습니다.

Queue 내에 있는 매서드 Enqueue()를 이용하여 인스턴스화 된 노트들이 배열에 생성이 되고, 사라질 땐 Dequeue() 매서드를 이용하여 메모리를 해제함으로써 힙에서 가비지 콜렉터가 활성화되는 것을 방지했습니다.

2. Player Cotroll

플레이어의 움직임은 코루틴을 사용하여 움직임을 조절했습니다.

가상의 큐브를 만들어 Raycast() 매서드를 이용, 가상의 큐브가 먼저 이동 및 회전을 이룬 후 이동 가능한 플레이트가 있는지 확인한 후 가상 큐브가 실행한 회전과 이동 거리를 목표로 두어 플레이어가 그 자리로 이동할 수 있도록 제작했습니다.

목표 거리까지의 이동거리 계산은 SqrMagnitude() 매서드를 통해 제곱근을 구하여 계산했습니다.

Player Controller 스크립트를 빈 오브젝트에 담은 후 움직일 큐브를 자식객체로 담았기 때문에 GetComponentInChildren() 매서드를 통해 Cube 오브젝트에 담긴 Rigidbody 컴포넌트를 선언 후 이동 가능한 플레이트가 없을 경우 useGravity를 체크, 큐브가 추락할 수 있도록 설계했습니다.

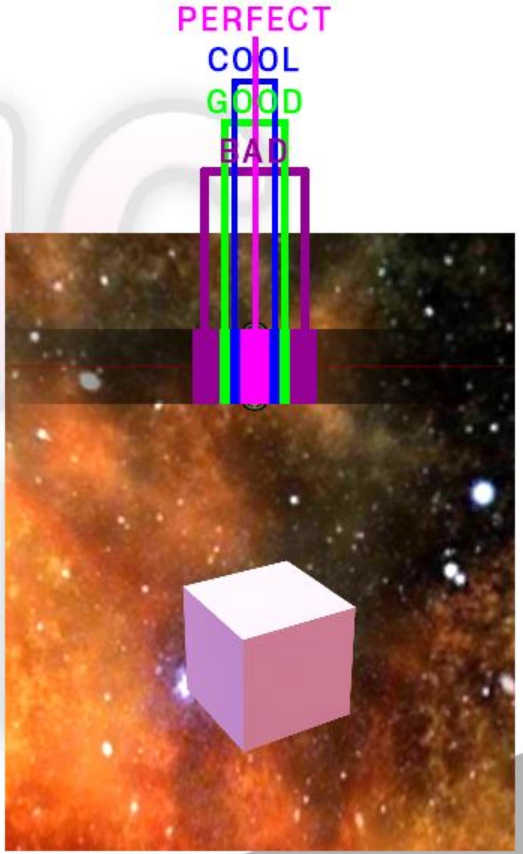
3. 노트 판정

노트 판정은 List를 이용하여 노트가 지나갈 때 판정범위에 있는지 확인할 수 있도록 했습니다.

타이밍 박스는 UGUI의 Image를 사용하여 월드스페이스가 아닌 캔버스 내부에서 좌표가 작동하므로 localPosition를 통해 Perfect, Cool, Good, Bad를 구분하게 제작했습니다.

판정이 들어갈 가운데 timingRect라는 오브젝트를 만들어 timingRect의 x좌표를 기준으로 판정처리 될 범위의 최소값, 최대값을 구분했습니다.

노트 판정이 정상적으로 될 시에 오브젝트 풀로 노트를 바로 이동시키는 게 아닌 오브젝트의 활성화 상태를 체크하여 이미지를 비활성화 시켜 노트의 등장 주기를 일정하게 유지했습니다.



HELLTAKER

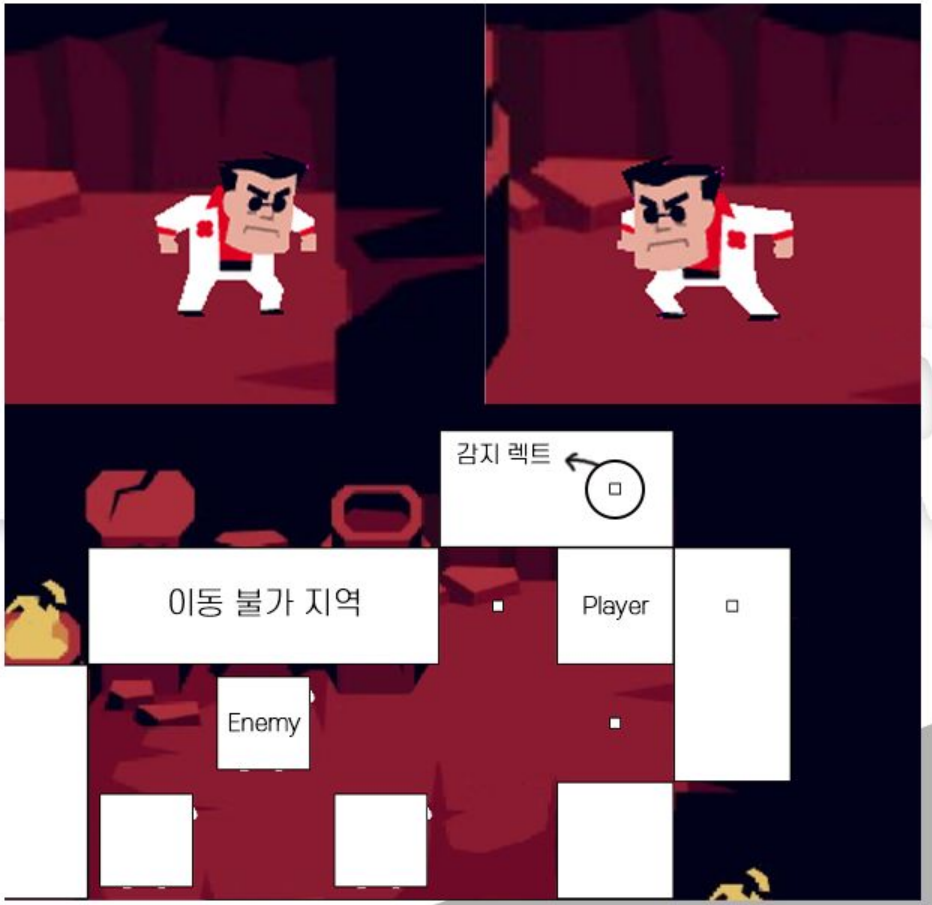
영상링크 : https://youtu.be/l4TEodZUKTA
GitHub : https://github.com/mmmdong/Helltaker1

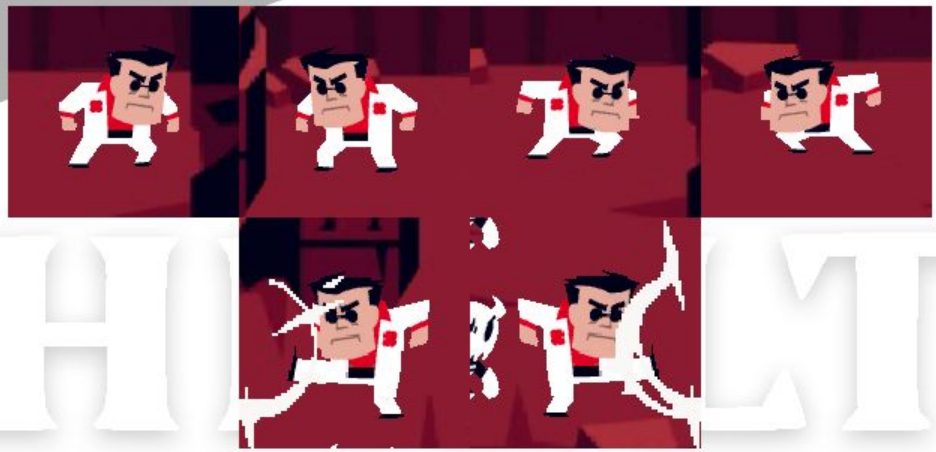
1. 이동 및 상호 작용

각 방향키를 이용하여 이동 및 속도 배지어 효과를 위해 마찰 가속력을 실수로 입력하여 자연스러운 움직임을 표현하였습니다.

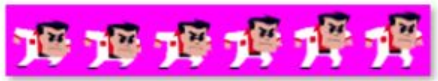
이동 시에는 움직이는 상태를 Bool로 선언하여 다른 방향키 입력이 불가하게 제작했습니다.

또한 상자 혹은 해골을 밀어내기 위해 감지 렉트를 추가하여 충돌처리를 정리했습니다.





dash_left.bmp



dash_right.bmp

idle_left.bmp

idle_right.bmp

kick_left.bmp

kick_right.bmp

2. 상태 및 이미지

'StageBase' 라는 헤더 파일에 정보를 입력한 후 각각의 스테이지 별로 'StageBase'를 참조하였습니다.

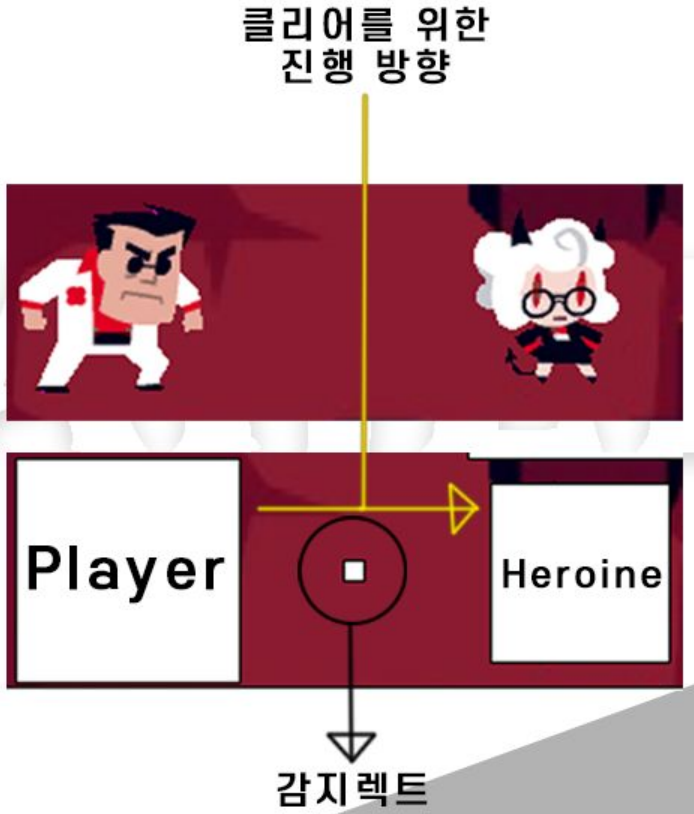
베이스 안 Enum문을 사용하여 캐릭터의 상태를 나누어 놓았고, 상태별로 이미지를 찾아 적용시켰습니다.

이미지 재생은 먼저 플레이어를 구조체로 만들어 재생할 이미지를 넣은 후 렌더할 애니메이션, 입력할 애니메이션을 따로 사용하여 상태별 프레임 차이를 극복했습니다.

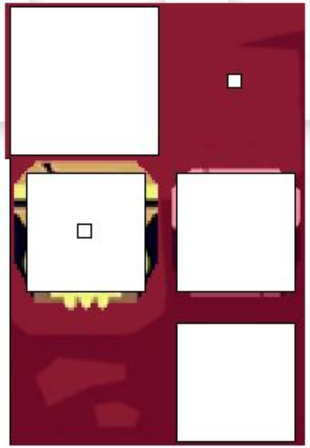
그리고 setFPS() 함수를 이용하여 애니메이션의 재생속도를 조절하였습니다

3. 게임 클리어

플레이어는 움직일 수 있는 이동 및 상호작용 횟수가 정해져있고 정확한 길을 통해서만 클리어에 도달할 수 있습니다.
상호작용을 감지할 수 있는 'inter' 렉트가 히로인 렉트에 충돌하면 클리어하면 clear 변수가 true로 전환되며,
'playGround'에서 setStage() 함수를 통해 스테이지의 초기값을 불러올 수 있도록 했습니다.
스테이지는 모두 'playGround'에서 최종 렌더되며,
구분은 switch문을 이용해 총 3개의 스테이지로 구성했습니다.
clear는 stageBase 헤더에서 정보를 받아올 수 있도록 getClear() 함수를 통하여 가져왔습니다.



HELLTAKER



4. Key

스테이지 2와 스테이지 3에서는 게임의 난이도를 위해 Key와 Box를 추가하여 key를 습득하지 못하면 Box가 막고 있는 길을 지나가지 못하도록 제한했습니다.

RIVER CITY GIRLS

영상링크 : https://youtu.be/hQ6omqxO-3c
GitHub : https://github.com/R1G4/Z-order

1. LoadScene

씬의 사운드는 fomd를 이용하여 싱글톤화 된 'SOUNDMANAGER'를 이용하여 addSound() 함수를 통하여 카운드를 추가. Play() 함수를 이용하여 재생했습니다.

각탭은 bool값을 통해 선택이 되었는지 판단하였고 탭이 선택될 때마다 false 에는 off 이미지, true 에는 on 이미지가 렌더링 되도록 했습니다.

파일A_off.bmp

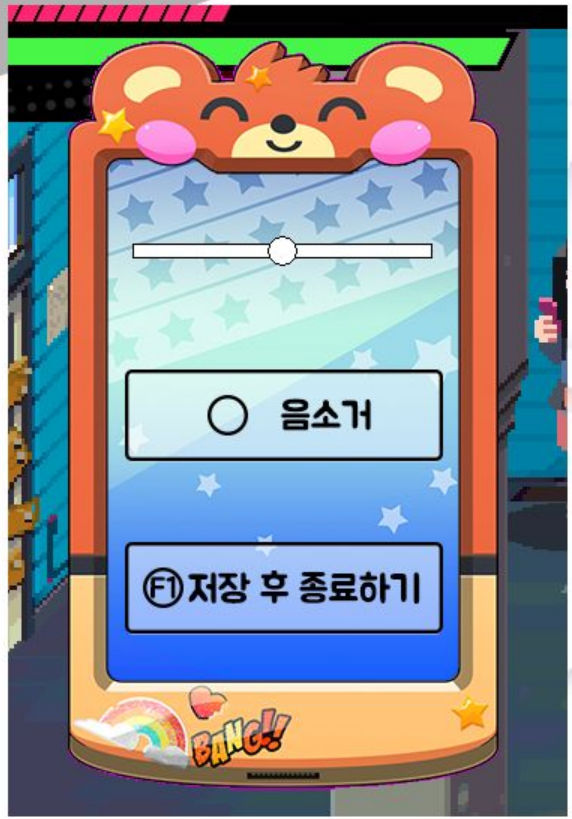
파일A_on.bmp

파일C_off.bmp

파일C_on.bmp

파일B_off.bmp

파일B_on.bmp



2. 옵션창

'M'키를 눌러 메뉴창을 호출합니다.

메뉴창의 호출을 bool값을 이용해 구분하였고, 호출된 상태에서 'F1'키를 눌러 현재 씬의 상태 및 캐릭터 체력상태를 저장하여 LoadScene으로 화면을 전환합니다.

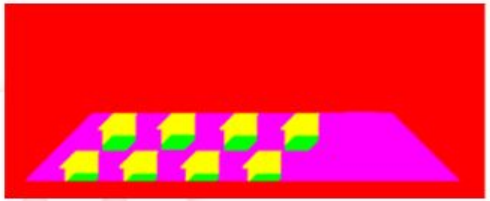
SOUNDMANAGER의 setVolunme() 함수를 통해 볼륨을 조절할 수 있고, 볼륨 조절의 정도를 winAPI의 좌표값에 따라 조절할 수 있도록 설정했습니다.

사각형 렉트를 제작해 0 ~ 100까지 조절할 길이를 정하고 위치의 X좌표에 볼륨상태(0~255)를 계산하여 원형 렉트를 통하여 조절할 수 있습니다.

3. Stage

각각의 스테이지의 충돌처리는 충돌처리될 부분의 이미지를 색으로 표현하여 충돌처리했습니다.

스테이지에 등장하는 기둥들은 캐릭터를 가릴 수 있어 기둥 이미지 사이즈에 맞는 렉트를 제작하여 플레이어 렉트가 기둥렉트에 충돌하게 되면, 이미지의 알파값을 낮춰 반투명하게 보이도록 처리했습니다.



The Witch's House

영상링크 : https://youtu.be/ZwuDELZ-pmE
GitHub : https://github.com/R1G4/witch-s-house

1. 정원

맵툴을 통해 필요한 좌표에 collider 속성을 넣었고 Z-order를 위해 플레이어 위에 정원의 나무가 나오도록 렌더 순서를 바꿨습니다.

각 collider 속성을 가진 타일들은 캐릭터의 상태에 따라 충돌시 이동하지 못하도록 제어했습니다.

마찬가지로 맵 이동시 캐릭터의 상태에 따라 스테이지별 시작지점을 지정했습니다. 맵 이동은 각 스테이지의 이동구간에 렉트를 만들어 렉트 충돌을 통해 이동하도록 제작했습니다.

2. Sound Stage

각각의 스테이지 별로 방을 두었고 방에 있는 모든 퍼즐을 풀어야 스테이지를 클리어할 퍼즐을 얻게 됩니다.

bool값이 씬이 바뀌며 초기화 되는 점을 방지하기위해 'STAGEMEMORYMANAGER' 클래스를 만들어 싱글톤화 한 후에 스테이지 외적으로 bool값을 저장할 수 있도록 했습니다. 저장된 bool값은 getter, setter 함수를 통해 스테이지에 불러와 조절할 수 있도록 설정했습니다.

이를 통해 방 별로 bool값에 따라 출력될 다이얼로그, 선택지를 따로 두어 클리어의 난이도를 조절했습니다.