기술 문서

김종혁
Phone: 010 - 3009 - 8396
mail: zxxd23@naver.com
Github : https://github.com/rebesian
Youtube :
https://www.youtube.com/channel/UCPlzliT7DvLfSpJ-ncfoEVQ

Portforlo

파이어 엠블렘 : 봉인의 검

개발 인원 : 1명

개발 기간 : 4주

개발 언어 : C++

개발 도구 : Visual studio
winApi
GitHub
SorceTree

링크:

Crypt of the NecroDancer

개발 인원 : 5명

개발 기간 : 7일

개발 언어 : C++

맡은 파트 : Enemy

개발 도구 : Visual studio
winApi
GitHub
SorceTree

링크:

Ninja baseball batMan

개발 인원 : 5명

개발 기간 : 7일

개발 언어 : C++

맡은 파트 : Player , Object

개발 도구 : Visual studio
winApi
GitHub
SorceTree

링크:

Lost Vikings 2

개발 인원 : 1명

개발 기간 : 7일

개발 언어 : C++

개발 도구 : Visual studio
winApi

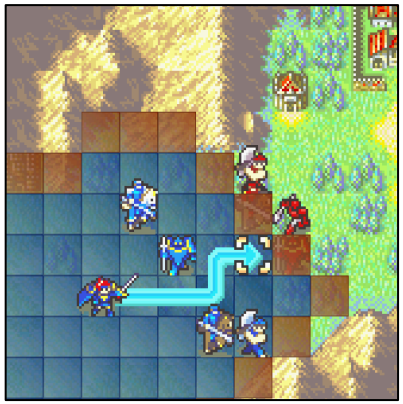
링크:

Player 턴

파이어 엠블렘은 턴제 방식으로 게임이 진행됩니다.
유저는 Player유닛을 하나씩 선택하여 조작 해야 하기 때문에 현재 게임 진행 상태를
총 4개의 bool로 표현하였습니다.
모든 플레이어 유닛의 행동이 끝나면 Enemy 턴으로 넘어갑니다.

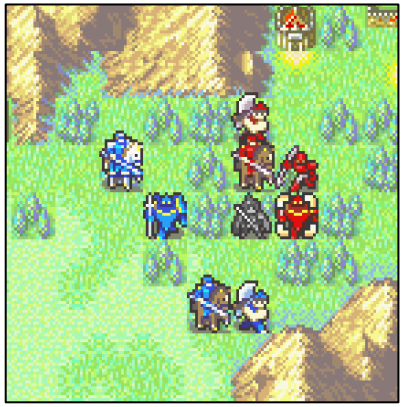
1. 유닛이 선택 되었는가?

2. 유닛이 이동 상태인가?



3. 유닛이 공격 상태 인가?

4. 유닛의 행동이 끝났는가?

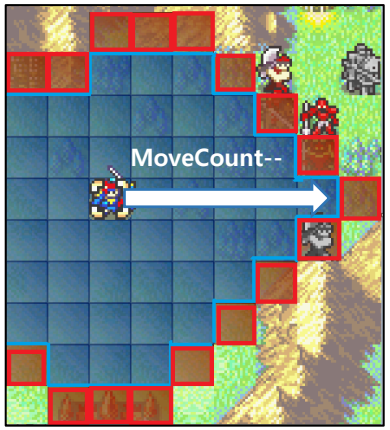


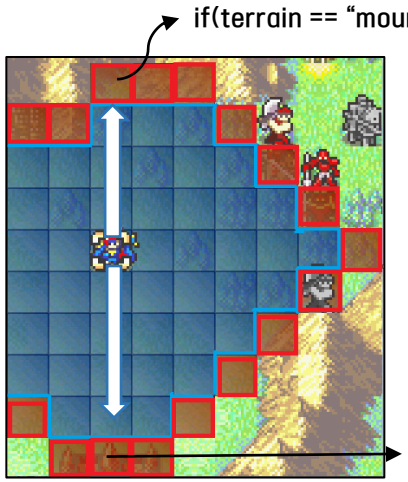
유닛의 이동범위

유닛이 가지고 있는 고유 이동범위를 표현 하기위해 A*알고리즘을 수행하는 TotalList부분에 Move속성과 Attack속성을 부여 하여 유닛의 이동범위를 나타냈습니다.

: Move

: Attack





- 해당 유닛의 좌표를 기준으로 Move속성 연산을 수행합니다.

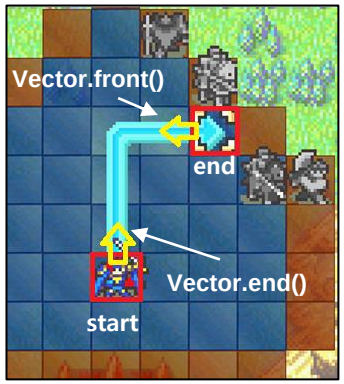
1. 상,하,좌,우
2. 1사분면, 2사분면, 3사분면 ,4사분면

- MoveCount = MoveRange 대입 MoveCount>0 일때만 Move속성 연산을 수행합니다.
- Move를 부여 했다면 MoveCount--연산을 수행합니다.
- For문으로 반복합니다.

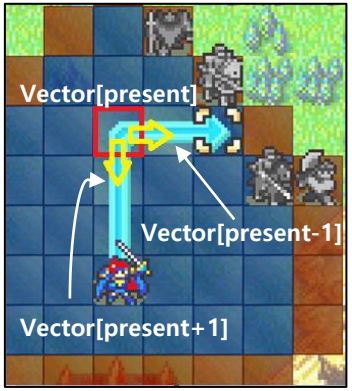
- 이동을 할 수 없는 지형에는 MoveCount를 0으로 설정하여 Move속성 연산을 끝냈습니다.

경로 구현

유닛이 이동상태 일 때 예상 경로를 표현하기 위해
A*알고리즘을 사용하여 예상 경로를 연산한 것을 화면에 이미지로 표현 하였습니다.
각각 경로의 이미지의 방향을 파악하고 예상 경로를 표현 하였습니다.



- start 타일은 Vector.back()원소의 (X,Y)을 비교하고 자신의 경로 이미지 방향을 결정합니다.
- End 타일은 Vector.front() 원소의 (X,Y)을 비교하고 자신의 경로 이미지 방향을 결정합니다.

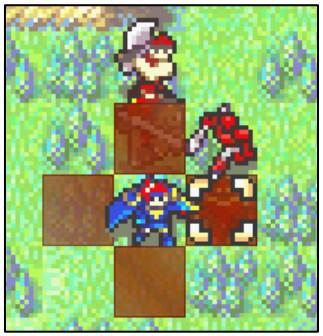


- StartTile과 EndTile이 아닌 다른 경로타일은Vector[present]을 현재 타일이라고 가정하고
  먼저 Vector[present – 1]원소의 (X,Y)을 비교한 다음,
  Vector[present + 1]원소의 (X,Y)을 비교하여 자신의 경로 이미지 방향을 결정합니다.

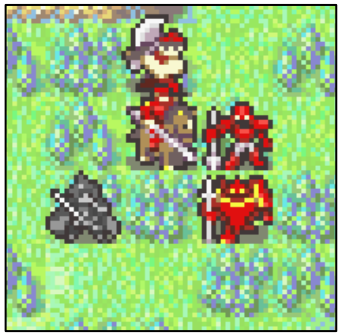
공격 대상 선정

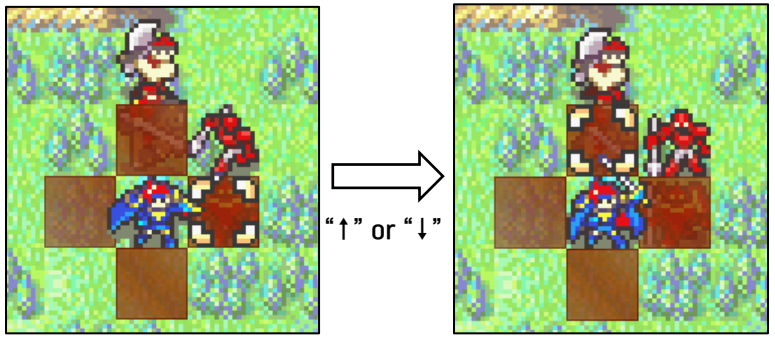
유닛이 이동을 끝냈을 때, 주변을 체크하여 적 유닛이 있다면 공격상태가 됩니다.
공격 범위에 체크된 적 객체를 타겟팅 해주었습니다.

- 주변에 적이 있는 경우



- 주변에 적이 없는 경우





- 유닛의 상,하,좌,우에 적 유닛이 있는 확인하고 있다면 적 유닛의 좌표정보를 Vector에 넣습니다.
- Vector.size()>0 이라면 isAttack상태가 됩니다.
- Vector.size()<= 이라면 used상태가 됩니다.
- 위,아래키를 누르면 Vector에서 적 좌표를 접근자로 가져와 커서 좌표에 대입해, 커서 위치가 바뀝니다.

Enemy 턴

Enemy 턴일 때 전체 Enemy객체에서 한 유닛 씩 차례로 플레이가 진행됩니다.
Enemy 어트랙션(이동,공격,행동종료)순으로 AI를 구성하였습니다.
모든 Enemy 유닛의 행동이 끝나면 Player 턴으로 넘어갑니다.

• Enemy 이동상태

• Enemy 공격상태

• Enemy 행동종료상태

Enemy AI는

1. 랜덤으로 Player 한 유닛에게 이동합니다.

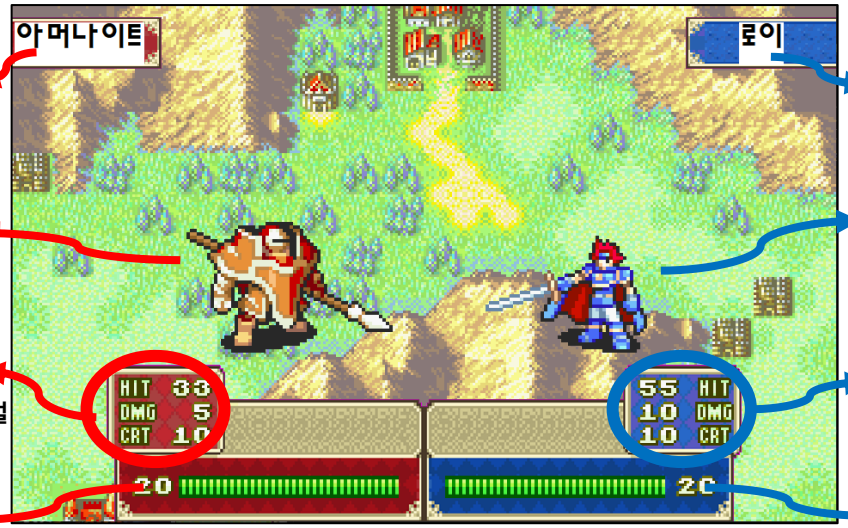
2. Player가 바로 주변에 있다면 공격합니다.
   주변에 Player 유닛이 여러 명 있다면 랜덤하게 선택합니다.

3. Enemy 유닛이 이동과 공격이 끝났다면 행동종료상태가 됩니다.

* Enemy도 이동범위가 있으며,
A*알고리즘으로 만들어진 경로로 이동하면서
이동하려는 타일이 Move속성이 아닐 경우
이동을 끝냅니다.

Battle Scene

Player 유닛과 Enemy 유닛의 전투가 진행 되는 씬 입니다.
서로 공격 대상을 선택하면, Battle Scene클래스에서 유닛의 battle의 필요한 정보들을 접근자로 받은 다음
유닛 마다 지정된 액션을 수행하고, 전투가 끝나고 난 뒤, 전투를 수행 한 유닛의 Hp를 반환하고 전투가 끝납니다.



Enemy 유닛 이름

Enemy 유닛 공격 모션

Enemy 유닛 명중률, 공격력, 크리티컬

Enemy 유닛 Hp바

Player 유닛 이름

Player 유닛 공격 모션

Player 유닛 명중률, 공격력, 크리티컬

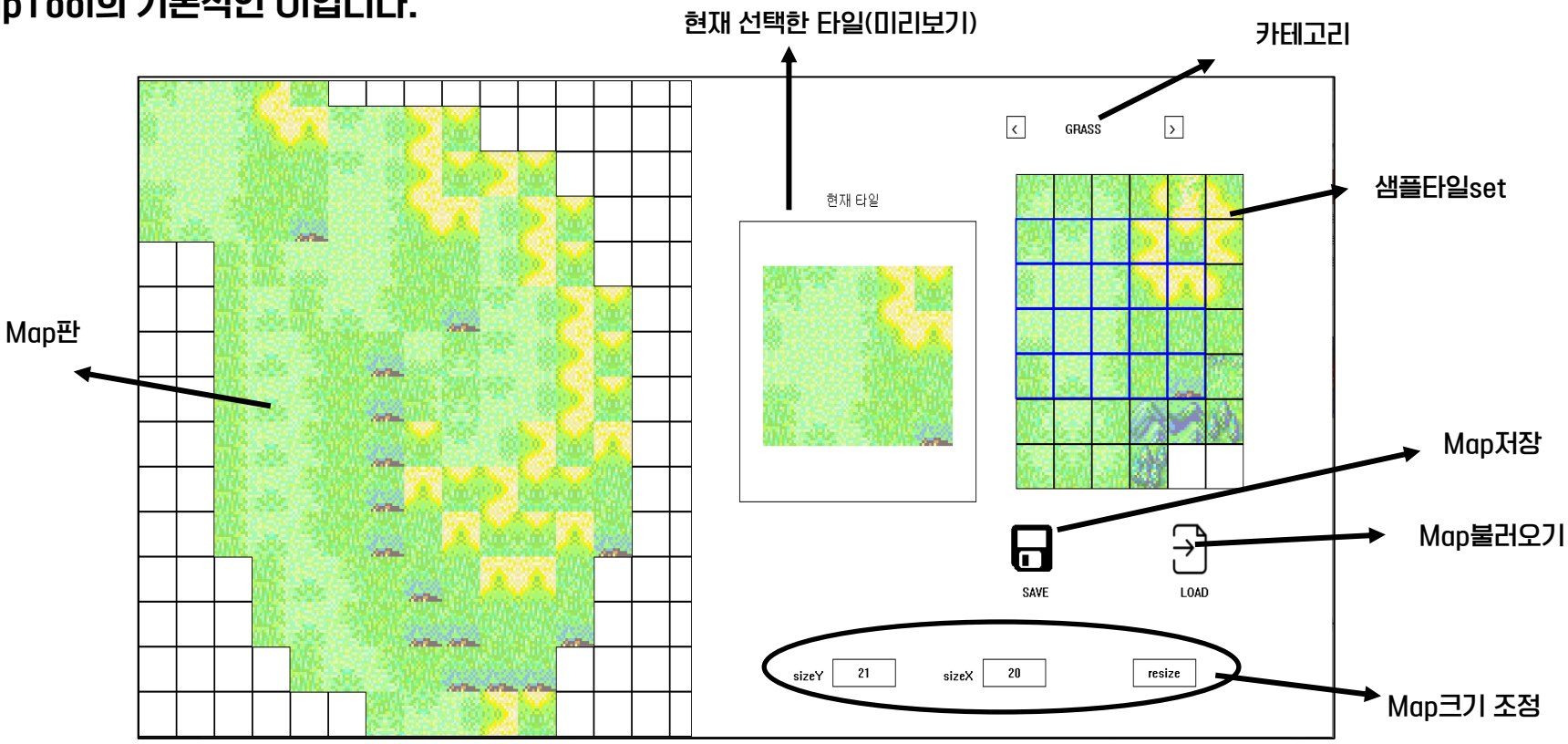
Player 유닛 Hp바

- 공격 모션 중 타격하는 모션일때만 Hp가 감소하게 구현 하였습니다.

- Hp바는 (유닛의 현재 체력/유닛의 전투 전 체력) x Hp바 이미지크기 공식으로 구현하였습니다.

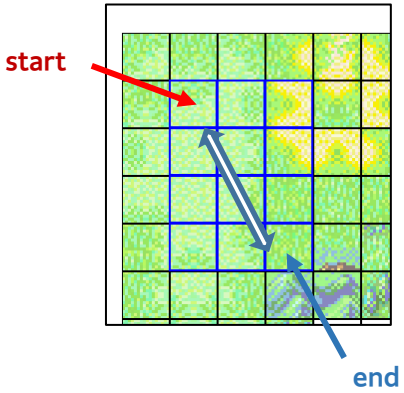
MapTool - UI

MapTool의 기본적인 UI입니다.

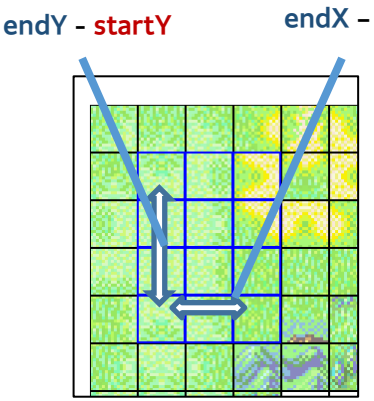


MapTool – 타일 선택,그리기

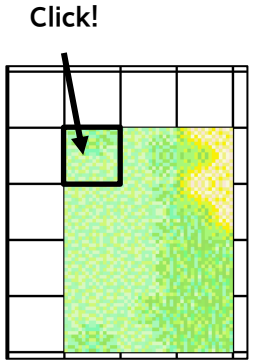
TileMap을 그리기 위한 MapTool의 주요 기능입니다.
MapTool의 UI이나, Map타일의 상호작용 방법은 PtInRect()함수를 이용하여 작동하게 했습니다.



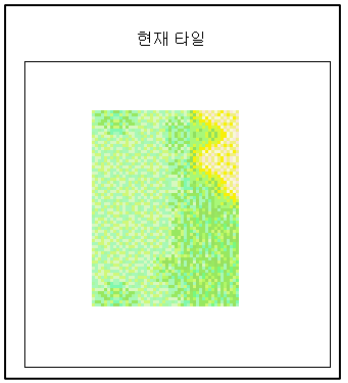
- 우 클릭하여 샘플타일을 선택해서 시작 지점의 좌표를 지정합니다.
- 우 클릭 한 채로 드래그해서 샘플타일의 끝 지점의 좌표를 지정합니다.



- EndX,Y – StartX,Y연산을 하여 선택한 타일의 범위를 구합니다.



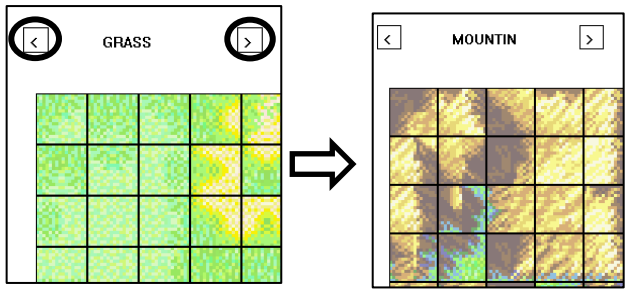
- Map판에 좌 클릭하여 클릭한 좌표를 기준으로 선택한 샘플타일set의 프레임 Index를 부여합니다.

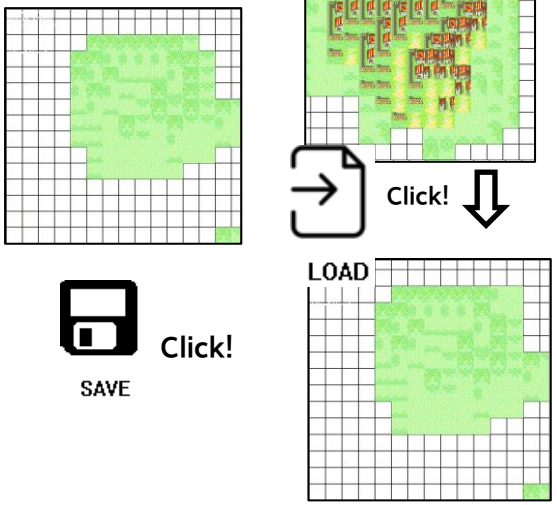


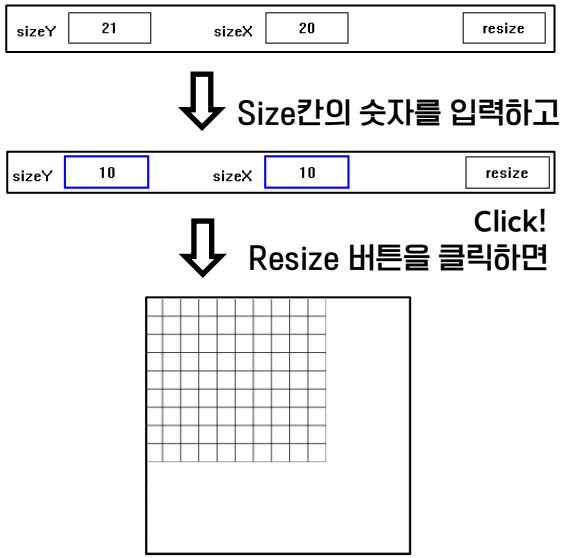
- 현재 타일에도 같은 방법으로 미리보기 이미지를 띄웁니다.
- 현재 타일 표시 RECT의 중점 좌표를 기준으로 가운데 정렬을 하였습니다.

MapTool – 부가 기능

Map을 그리는 것 외의 부가적인 기능입니다.
“카테고리 변경”, “저장,불러오기” , “Map크기조정“에 대해 설명 하겠습니다.







<, >라고 표시된 RECT를 클릭하면 카테고리가 바뀝니다.

WriteFile()을 사용해서 Map을 저장합니다.

ReadFile()을 사용해서 Map을 저장합니다.

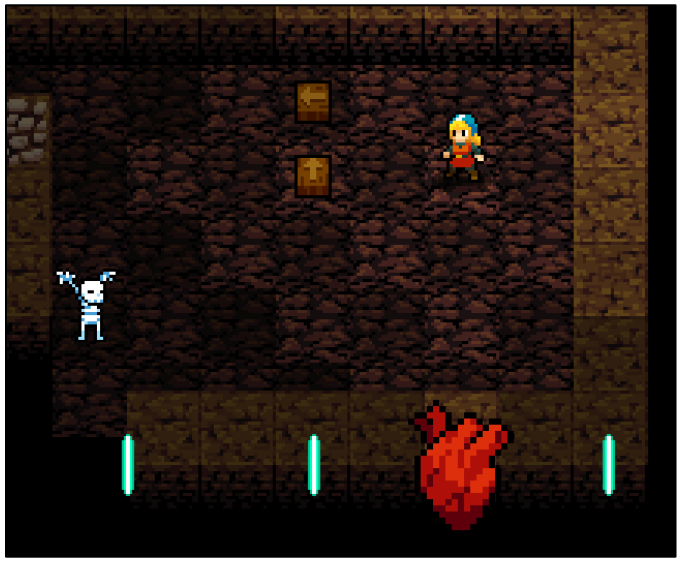
이중백터로 구성된 Map크기를 각 X,Y축에 push_back()와 erase()함수를 이용하여 Map의 크기를 조정합니다.

- 샘플타일Set 선택
- 저장 & 불러오기
- Map크기조정

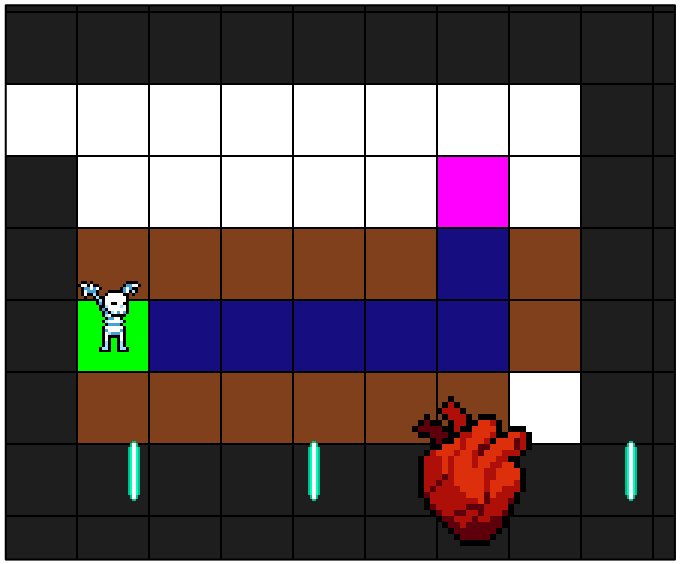
Enemy 기본패턴 - 1

Crypt of the NecroDancer의 Enemy의 기본패턴은 Enemy유닛의 탐사범위에 Player가 들어온다면 A*알고리즘을 사용하여 Enemy와 Player까지의 경로를 만듭니다.

- 기본 Enemy의 배치



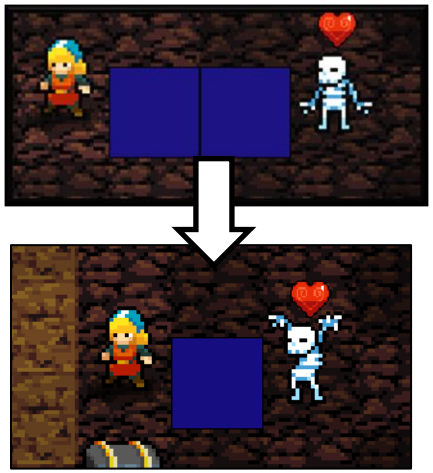
- A* 알고리즘 클래스의 표기



Enemy 기본패턴 - 2

A*알고리즘을 사용하여 경로가 만들어 지고 난 후,
Enemy를 움직이게 경로를 따라가서 Player를 공격하도록 구현 하였습니다.

- Enemy 기본 이동

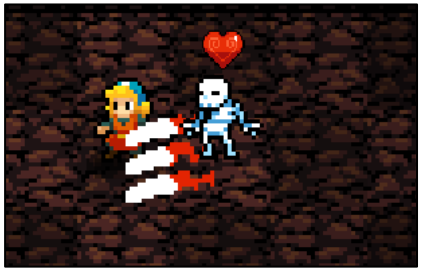


- 경로가 만들어지면 유닛마다 정해진 박자에 맞춰 한 칸 씩 이동 합니다.
- 이동하고 경로에 pop_back()을 합니다.
- 박자는 if(현 시간 – 전 시간 >= 박자간격)으로 처리 하였습니다.

- 장애물 이동 방지

- 이동하기전에 경로의 마지막 원소 좌표를 참조하여 장애물 유무를 파악합니다.
- 장애물이 있다면 앞으로 이동하지 않고 제자리 점프를 하게 구현 하였습니다.

- 공격



- 경로Vector.size()가 0이면 주변에 Player가 있는지, 상,하,좌.우로 판단하고 Enum문 변수에 방향을 지정합니다.
- 이동 박자에 맞춰서 해당 방향을 공격합니다.

Enemy - Skeleton

Skeleton은 Crypt of the NecroDancer의 가장 기본적인 패턴을 가지고 있는 Enemy 유닛 입니다.
“Skeleton”, “GreenSkeleton”, “BlackSkeleton” 이렇게 3종류의 유닛이 존재합니다.
3종류의 차이점은 Hp가 다르다는 점 입니다.
Skeleton고유의 특징은 Skeleton의 탐사범위에 플레이어가 들어오면 팔을 드는 모션을 취합니다.

- Skeleton의 탐사 전

- Skeleton의 탐사 후

Skeleton
Hp1

GreenSkeleton
Hp2

BlackSkeleton
Hp3

Enemy - Ghost & Mimic

Ghost와 Mimic의 패턴은 탐사범위가 바로 자신의 상,하,좌,우 1칸 앞에 플레이어가 있다면
모션이 바뀌고 , 플레이어를 1박자에 맞춰서 따라갑니다.
만약 탐지가 된 모션이 되지 않으면, Player의 공격을 받지 않게 하였습니다.

- Ghost 탐사 전

탐지가 되기 전엔
AlphaRender의 Alpha값을
100으로설정합니다.

- Ghost 탐사

탐지가 되고 나면
Alpha값을 255로
설정합니다.

- Mimic 탐사

탐지가 되기전엔 정해진 프레임
전 까지 모션을 진행합니다.

- Mimic 탐사 후

탐지가 되었다면 원래
프레임까지 모션을 진행합니다.

Enemy - Monkey

Monkey의 공격 판정은 직접적으로 Player의 Hp를 감소하게 하지 않습니다.
대신 Monkey는 붙어서 Player의 이동을 제한합니다. 이동을 제한 할 때, Mokey의 Hp도 늘어납니다.
이때 플레이어가 박자에 맞춰 방향키를 누르면 Monkey의 Hp가 감소하고, 0이 되면 사라집니다.

- 잡기 전 Monkey

Monkey는 Player가 1칸 근처에 있다면 공격을 잡기를 시전합니다.

- 잡기 상태 Monkey

Monkey가 IsCatch == true 라면 플레이어가 박자에 맞춰 방향키를 누르면 Monkey의 Hp가 닳습니다.

- 잡기 후 삭제 된 Monkey

Monkey 체력이 0이 되면 삭제해줍니다.

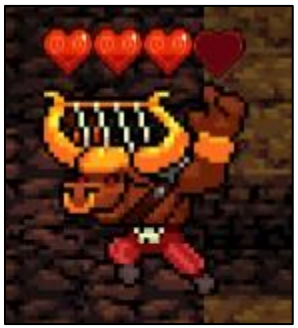
Enemy - Minotauros

Minotauros는 MiniBoss유닛이며 MiniBoss를 잡아야 보스방으로 갈 수 있게 하였습니다.
Enemy기본패턴을 사용하지만, MiniBoss유닛 답게 특수패턴이 존재합니다.

- 특수패턴 off

- Enemy기본 패턴을 사용합니다.
- 경로대로 이동할 때 이동 할 경로 좌표에 벽 Object가 있다면 벽을 부수면서 이동합니다.
- 2박자에 이동합니다.

- 특수패턴 On



- 플레이어 좌표와 같은 X축이나 Y축선상에 있다면 특수패턴(bool)이 true로 변환되어 실행됩니다.
- 경로가 지워지고 플레이어 또는 벽Object에 닿을 때 까지 한 방향으로만 이동합니다
- 1박자에 이동합니다.

- 플레이어 충돌

- 벽Object 충돌

- 플레이어에 부딪혔다면 플레이어에게 대미지를 입히고, 2박자간 스턴을 먹습니다.
- 벽Object에 부딪혔다면 해당 벽을 부수고 2박자간 스턴을 먹습니다.
- 스턴에 깬다면 특수패턴(bool)을 false로 변환하고 다시 Enemy기본패턴을 사용합니다.

Enemy - DeathMetal(Boss)

Boss유닛 DeathMetal입니다. DeathMetal는 총 4 페이즈로 나뉘여 있고, 현재 Hp에 따라 페이즈가 진행됩니다.

- 1페이즈 (HP ≥ 7)

- Enemy기본 패턴을 사용합니다.
- 경로 이동방향을 enum문으로 저장하여 자신이 방패를 들고 있는 방향일때 상황과 플레이어의 좌표를 비교하여 방패가 있는 곳은 Hp가 감소하지 않습니다.

- 2페이즈 (7 > HP ≥ 5)

- A*로 만들어진 경로로 이동하지 않고, 공격도 하지 않습니다. 그저 정해진 거리를 유지를 하기위해 움직입니다.
- 피해를 입으면 방의 모서리 지점 으로 워프합니다.
- 8박자마다 Skeleton을 소환합니다.

- 3페이즈 (5 > HP ≥ 3)

- 2페이즈의 움직임과 같습니다.
- 8박자마다 GreenSkeleton을 소환합니다.

- 4페이즈 (3 > HP ≥ 0)

- Enemy기본 패턴을 사용합니다.
- 8박자 마다 화염포를 자신의 양옆 타일을 벽에 닿을 때 까지 쏘는데, 벽에 닿을 때 까지 자신의 좌표 기준 양 옆 타일들의 좌표정보, 공격판정(bool), 이미지를 Vector에 넣어서 화염포를 구현했습니다.

Ninja Baseball BatMan

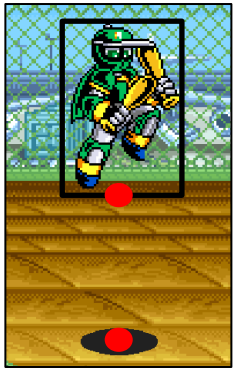
벨트 스크롤 액션

Ninja Baseball BatMan게임의 특징인 벨트 스크롤 액션을 표현하기 위해
좌,우,아래 이동은 Camera좌표 기점으로 넘어가지 않게, 위쪽 이동은 바닥 이미지에 맞게 이동을 제한했습니다.
또한 Z축의 역할을 해주기 위해 그림자를 사용하여 점프나, 위 아래 움직임을 Z축도 사용한 것 처럼 구현하였습니다.

- 왼쪽으로 나가지 못하게 Camera 좌표 기준으로 제한하였습니다.

- 기본 게임 화면

- 맵 이미지에 맞게 Y축의 이동을 제한 하였습니다.



- 점프는 캐릭터 Y좌표를 사용합니다.
- 착지는 캐릭터 아래쪽 좌표가 그림자 중점의 위치 아래로 내려가게 되면 Idle상태로 변환 합니다.

공격 모션 분기

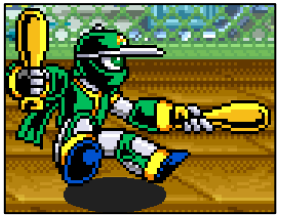
Ninja Baseball BatMan의 플레이어는 “상태패턴”을 통해 여러 공격상태를 정의 했습니다.
공격 상태마다 해당 프레임일때 지정 커멘드를 입력 받아 어떤 Bool변수가 true인지를 구분하여 상태를 변환하였습니다.

• 기본 Attack 상태

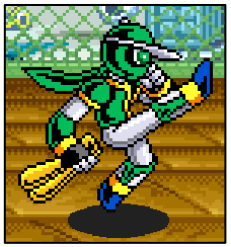
• 기본 Attack 상태

• FrontAttack 상태

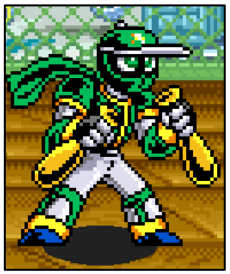
• Idle 상태



1. 그대로 공격 키를 누른다. (isAttack)
: Attack상태의 프레임Index를 증가 시켜
기본 Attack상태를 유지합니다.



2.-> && 공격키를 누른다. (isFrontAttack)
: 기본 Attack상태클래스를 Delete하고
frontAttack상태로 변환하였습니다.



3.아무것도 누르지 않는다. (isend)
: 해당 If(frameIndex == 2) 분기일때 ,
int변수 Count를 ++연산을 수행하고
Count>30의 조건이 되었을 때,
Idle상태로 변환합니다.

```
If(frameIndex == 2 )
{
  1.그대로 공격 키를 누른다. (isAttack)
  2.-> && 공격키를 누른다. (isFrontAttack)
  3. 아무것도 누르지 않는다. (isend)
}
```

대쉬

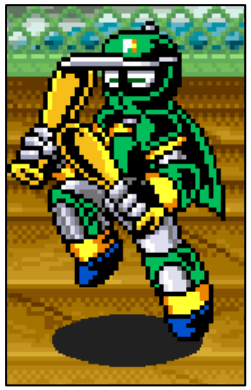
같은 좌,우 방향키를 연속으로 눌러서 대쉬 상태로 갈 수 있게 구현하였습니다.
좌,우 방향키를 한번 눌렀을 때, 지정된 시간내에 한 번 더 같은 방향키를 누르게 되면 대쉬상태로 하게 만들었습니다.

- Idle 상태

- Idle상태에서 왼쪽, 오른쪽 방향키를 누르면 Move상태로 들어갑니다.

→,← 방향키 입력

- Move 상태



- Move상태로 바뀌면 플레이어 클래스안의 (bool)isRun이 true값으로 바뀝니다.
- 플레이어 클래스에 (float)runtime+=0.1연산을 수행하여 일정 수치내에 방향키를 또 누르면, (bool)_run이 true로 바뀝니다.

같은 방향키 입력

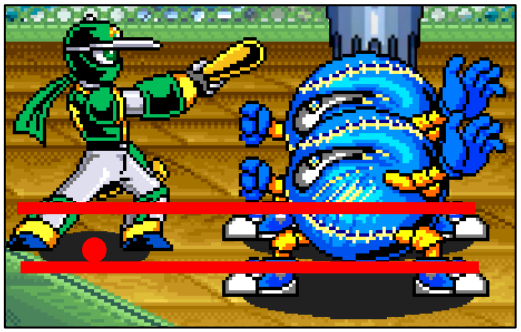
- Dash 상태

- Move상태에서 if(isRun && _run)를 만족한다면 Dash상태로 변환합니다.
- (bool)IsRun과 (bool)_run를 false로 설정합니다.

공격, 피해, Object 상호작용

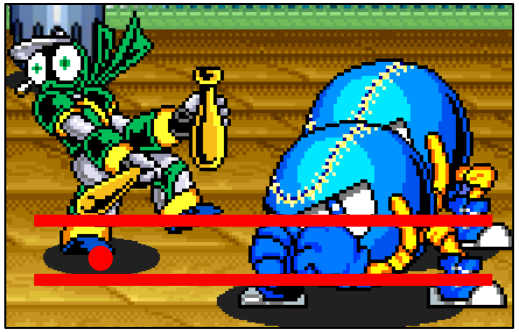
Ninja Baseball BatMan게임의 공격 ,타격 , 오브젝트 상호작용, 충돌처리를 하는 판단은 그림자 범위가 넓은 쪽에 캐릭터 or Object가 상대 그림자 범위 Y축 범위안에 들어와야 하는 동시에 공격판정의 Rect와 캐릭터 Rect가 충돌, Object의 Rect와 캐릭터의 Rect가 충돌해야 합니다.

- 공격 상황



- Enemy의 그림자 Y축 범위가 넓기 때문에 Player의 그림자 중점이 Enemy 그림자 Y축 범위 내에 있는지 먼저 판단합니다.

- 타격 상황



- Attack RECT충돌로 인해 상황이 발생됩니다.

- 오브젝트 상호작용

- Player 그림자 Y축 범위가 넓기 때문에 오브젝트의 그림자 중점이 Player 그림자 Y축 범위 내에 있는지 먼저 판단합니다.
- 지정된 커맨드로 Item과 상호작용 합니다.

Item – 쓰레기통

맵에 존재하는 아이템을 먹으려면 쓰레기통을 부숴야 합니다.
쓰레기통을 부수는 건 공격으로 부수며, 이는 Object 상호작용 방법과 같습니다.
부숴지면 파편이 나오면서 아이템이 나오는 연출을 하였습니다.

- 부숴지기 전

- 쓰레기통과 플레이어 공격판정Rect와 IntersectRect()가 된다면 쓰레기통 클래스의 (int)Count가 ++연산을 실행합니다.
- if((int)Count>3)이라면 부숴집니다.

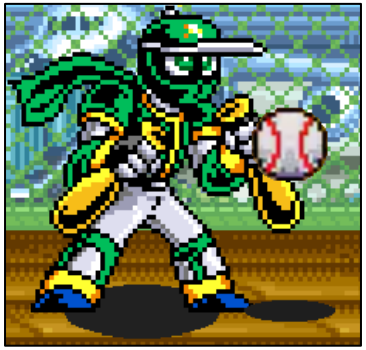
- 부숴진 후

- 부숴지면 현재 부숴진 쓰레기통의 (Int)present 번호와 대치되는 지정된 아이템의 (bool)IsAppear에 true 값을 주어 아이템이 출현합니다.
- 파편에 중력 값을 주어서 , 쓰레기통의 그림자 중점 아래로 내려가면 사라집니다.

Item – 야구공

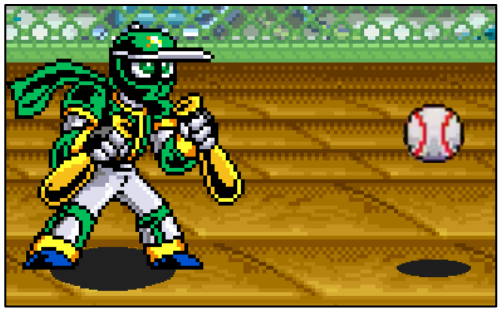
야구공은 Item 중 보조 무기 형식으로 사용합니다. 던져서 적에게 타격을 주기도 합니다.
야구공을 잡는 방법은 Object 상호작용과 같습니다.

- 야구공을 잡음



- 야구공을 잡으면 (bool)IsHold를 true값으로 변합니다.
- IsHold이 true일때 던지는 액션이 가능합니다.

- 야구공을 던짐



- 야구공을 던지는 커맨드를 입력하면 (bool)Isattack와 (bool)isFire가 true값으로 변하고, (bool)isHold는 false값로 변합니다.
- X축으로만 이동하고, Player 방향을 기준으로 합니다.
- 방향을 고정하기 위해서 (bool)isFire 추가 사용 하였습니다.

- 야구공과 에너미의 충돌

- Enemy가 야구공에 맞게 되면 야구공은 delete됩니다.
- 야구공이 Enemy와 충돌처리가 되지 않고, Camare상에 나가게 되도 delete 합니다.

Item - 회복 아이템

먹으면 회복이 되는 Item 입니다.
회복 아이템을 먹는 방법은 Object 상호작용 하는 방법을 기반으로 커맨드를 누르면
현재 Hp를 회복해줍니다.

- 회복아이템 먹기 전

- 회복아이템 먹고 난 후

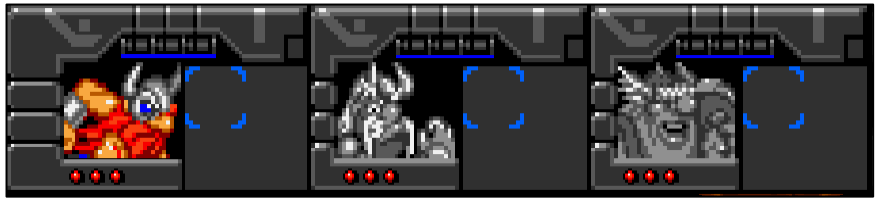
- 회복 아이템은 현재 Hp를 +1 증가 해줍니다.
- 현재 Hp가 Full이라면 Hp+1증가 하지 않습니다.

Lost Viking2

캐릭터 조작

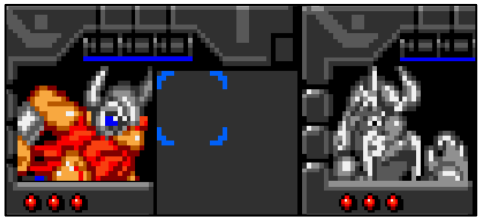
Lost Viking2 캐릭터마다 다른 특성을 가지고 있어, 한 캐릭터 씩 조작 하여 퍼즐을 풀어가는 게임입니다. 그래서 캐릭터를 한 명씩 조작 해야 하는 상황을 만들었는데 각 캐릭터마다 (Bool)Present를 true/false로 현재 조작중인 캐릭터를 표시합니다.

- 플레이어 표시 UI



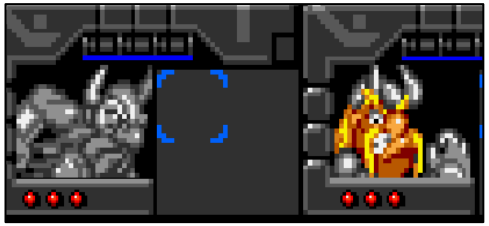
(bool)Present = true　(bool)Present = false　(bool)Present = false

- 현재 조작 가능한 캐릭터를 UI에 표시합니다.
- 조작가능한 캐릭터는 Bool값을 True로 설정합니다.



(bool)Present = true　(bool)Present = false

"Q", "E" 입력으로
현재 조작가능한 캐릭터를 변경

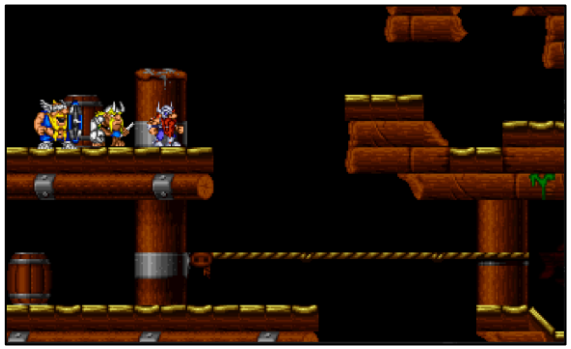


(bool)Present = false　(bool)Present = true

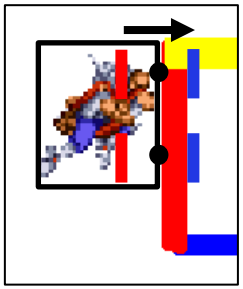
Lost Viking2

픽셀 충돌

Lost Viking2의 맵은 지형이 일정하지 않아서 GetPixel()함수를 사용 한 픽셀충돌로 지형을 구현 하였습니다.



• 기본 맵

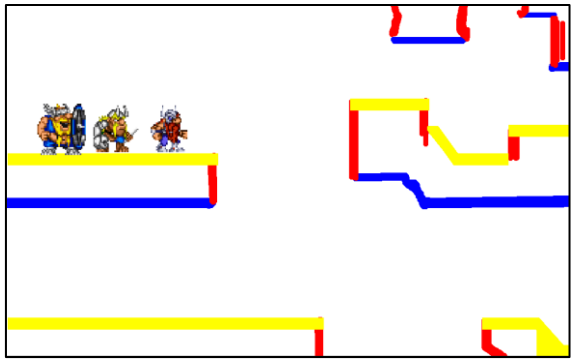


● : 탐사 기준점

: 탐사 시작점

: 탐사 끝점

→ : 탐사 진행방향



• 픽셀 충돌 맵

1. 탐사 기준점 좌표를 기준으로 시작점 부터 끝점 까지의 범위안에 GetPixel()를 사용해 특정 RGB값의 픽셀이 있다면 가장 시작점에 가까운 픽셀 좌표를 반환합니다.

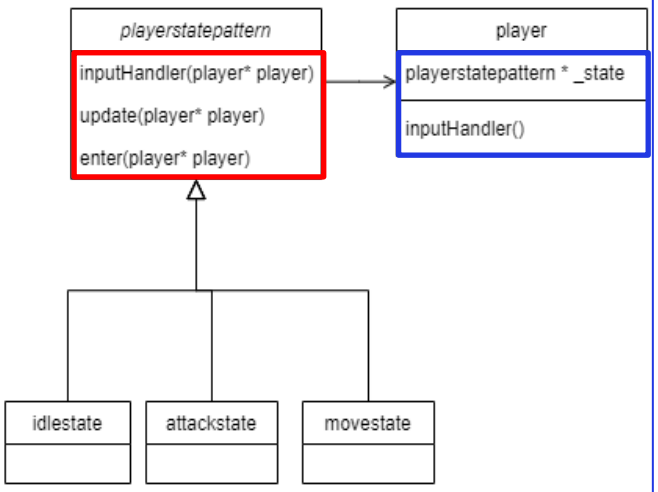
2. (반환된 좌표 – PlayerRect좌표) 한 만큼 PlayerRect좌표를 조정하여 바닥 , 천장, 벽을 넘지 못하게 이동을 제한합니다.

Lost Viking2

상태 패턴

**캐릭터들의 공통 된 행동 과, 캐릭터 마다 고유행동들을 표현하기 위해서 상태패턴을 사용했습니다.
기본 상태 인터페이스 클래스를 만든 후, 행동 상태들을 상속하여 모든 행동 상태들을 정의하고,
InputHandler()함수를 통해, 변환 할 상태클래스를 동적 할당하여 캐릭터의 상태를 구현하였습니다.**

- Input Handler() : 상태 클래스마다 다른 상태 이전 조건을 정의하고, 조건을 달성하면 해당 조건에 맞는 상태 클래스를 반환합니다.
- update() : 해당 상태의 행동을 수행합니다.
- enter() : InputHandler()로 새로운 상태클래스로 이전 되었을 때 한번 실행됩니다.



- (statePattern)* _state : 상태클래스를 받아 올 변수입니다.
- InputHandler() : 플레이어에 정의된 InputHandler는 3가지 형태로 동작합니다.

1. _state변수에 할당 된 상태클래스가 있는지 체크합니다. nullptr이면 체크하지 않습니다.
2. 체크 되었다면 이전 상태 클래스를 delete하고 새로운 상태 클래스를 _state에 대입 해줍니다.
3. enter함수를 실행하여 해당 상태의 데이터들을 초기화 해줍니다.

Lost Viking2

갈고리 액션

Baleog의 고유 액션으로 갈고리를 이용한 공격 모션 중 , 갈고리가 해당 액션 오브젝트에 충돌하게 되면, 밧줄에 매달리는 액션이 실행됩니다. 삼각함수로 구현 하였습니다.

1. 갈고리 손과 액션 오브젝트의 IntersectRect()가 발생하면, 공식을 사용해서 각도와 거리를 구합니다.
2. 구해진 각도와 거리를 이용해 고리부분과, 캐릭터 좌표를 설정 합니다.
3. 물리법칙을 적용합니다 좌, 우 방향키를 누르고 있으면 진자 운동을 하게 되고, 누르고 있지 않으면 서서히 속도가 줄게 됩니다.

- 공식

- angle : (플레이어 중점 X좌표 - 오브젝트 Rect중점 X좌표) /(플레이어 중점 Y좌표 – 오브젝트 Rect중점 Y좌표)
- Distance : [(갈고리손 X좌표 – 오브젝트X좌표)² + (갈고리손 Y좌표 – 오브젝트Y좌표)²)]½;
- 캐릭터 or 고리부분들 X좌표 : 자신의 앞쪽 갈고리 X좌표 + cosf(angle) * distance
- 캐릭터 or 고리부분들 Y좌표 : 자신의 앞쪽 갈고리 Y좌표 + sinf(angle) * distance

- 물리법칙

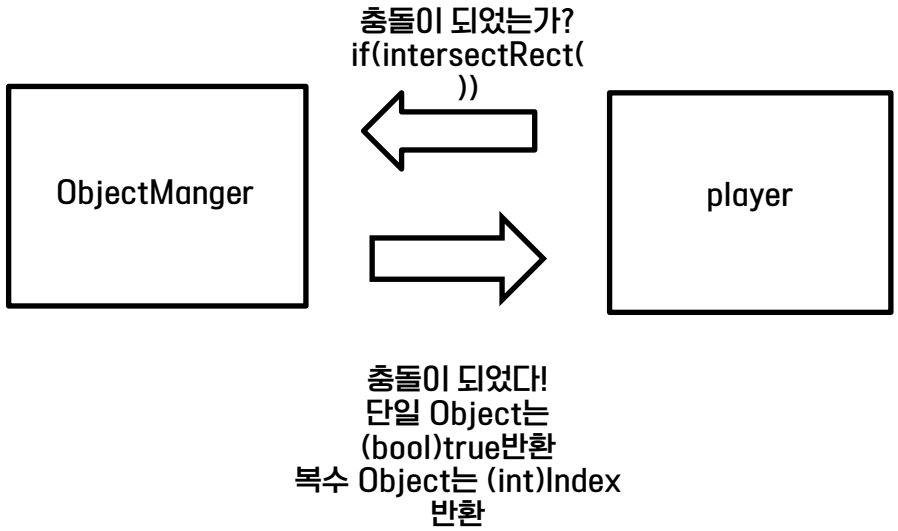
- 가속도 = (중력(0.08) / distance) * cosf(angle);
- 속도 += 가속도
- 속도 *= 0.9f (속도를 감소하기 위함)
- angle = 속도

- 갈고리 액션

Object 상호작용

Object의 상호작용은 모든 Object들을 관리는 ObjectManager 클래스를 두어 관리를 하였고,
Player 클래스와 전방선언을 통한 순환 참조 구조를 해결하여 충돌체를 이용한 축 충돌로 처리하였습니다.
캐릭터와 Object가 Rect충돌판정이 되었다면, Player클래스 내부의 대응하는 변수에 반환 받게 됩니다.

- 사다리와 캐릭터가 Rect충돌 된 모습

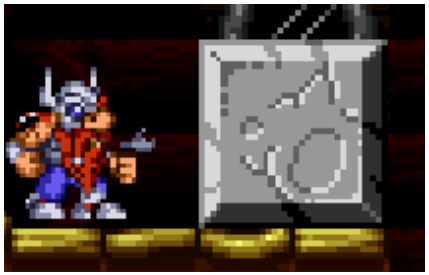


Lost Viking2

Object - 움직이는 벽

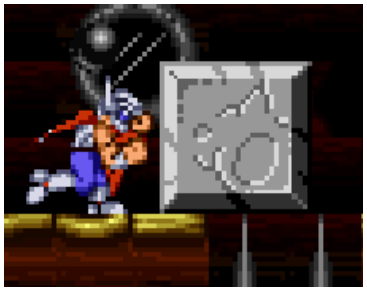
벽을 움직여서 길을 만들게 하는 오브젝트 입니다.
해당 Object 는 단일로 만 존재 하기 때문에 Player클래스 내의 대응하는 변수에 신호를 반환합니다.
벽 Object 위쪽에 올라가 있을 때를 대비해서, Player클래스에게 벽의 Rect정보도 같이 반환합니다.

- 벽Rect 충돌 전



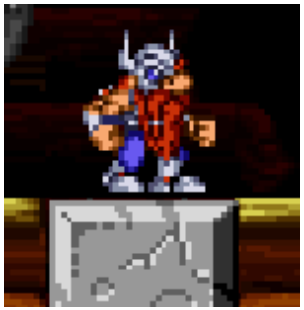
- 캐릭터와 벽 Rect가 충돌 했는지 상시 확인합니다.

- 벽Rect 충돌



- Object와 충돌 되었다면 Player클래스에 벽 Object 의 Rect정보를 반환합니다.
- 캐릭터가 벽과 충돌 한 상태에서 캐릭터Rect에 밀린 만큼 objectManger클래스 내에서 SetX()함수를 사용해 벽을 이동하게 합니다.

- 벽Rect 위에 있을 경우



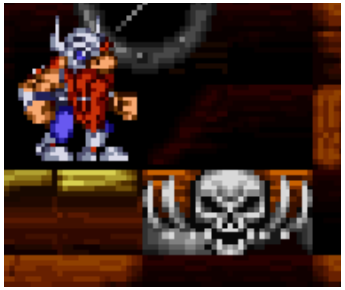
- 벽 Obejct 를 밟고 지나갈 수 있어야 하기 때문에 , 캐릭터 와 충돌 된 상황이 top쪽이라면, Player클래스 쪽에서 반환 받은 Rect 정보를 활용하여 캐릭터의 좌표를 벽 Obejct 위쪽으로 조정해줍니다.

Lost Viking2

Object - 엘리베이터

위, 아래방향키를 누르면 캐릭터를 위, 아래로 옮겨주는 Object 입니다.
해당 Object는 복수 Object 이기 때문에, 충돌 판정이 난 Object 배열Index를 반환합니다.
항상 캐릭터가 엘리베이터 위에 있어야 하기 때문에 Player클래스에게 엘리베이터 Rect정보도 같이 반환합니다.

- 엘리베이터 Rect충돌 전



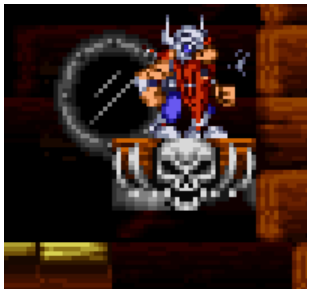
- 캐릭터와 엘리베이터 Rect가 충돌 했는지 상시 확인합니다.

- 엘리베이터 Rect충돌 후

- Object와 충돌 되었다면 Player클래스에 엘리베이터의 Rect정보와 엘리베이터의 Index를 반환합니다.
- Player클래스내의 엘리베이터 Rect정보로 캐릭터의 좌표를 위쪽으로 조정해줍니다.

- 엘리베이터 작동

“↑” or “↓”



- 위,아래 방향키를 누르면, 반환 받은 엘리베이터 Index정보를 가지고 캐릭터가 타고 있는 엘리베이터를 SetY()함수로 조작하게 됩니다.
- 마찬가지로 엘리베이터가 작동하면서 Player클래스 내의 엘리베이터 Rect정보로 캐릭터의 좌표를 위쪽으로 조정해줍니다.