Yeungnam University

Jumplime

Design Document

Student No	22112032
Name	김가빈
E-mail	rkqls4764@naver.com

[Revision history]

Revision date	Version #	Description	Author
06/10/2023	2.00	First Draft	김가빈

= Contents =

1. Introduction

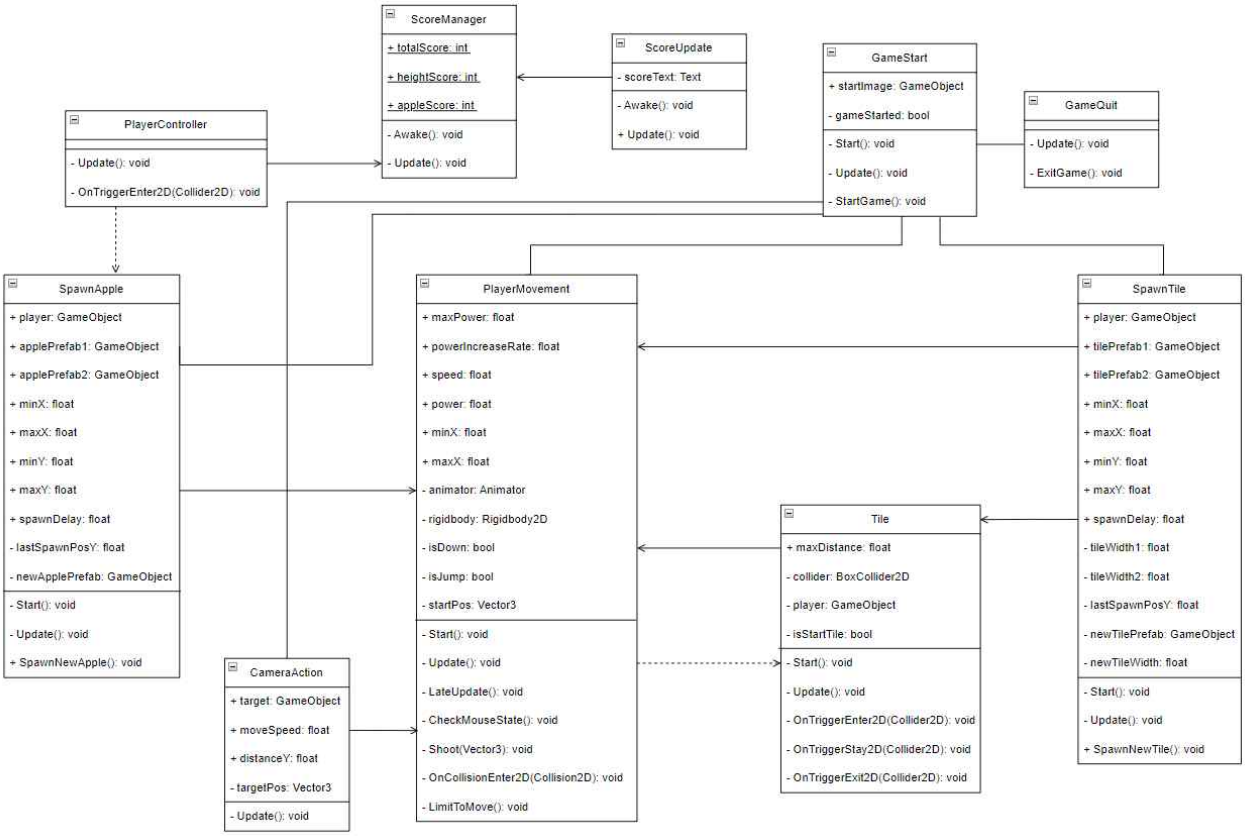
일상생활에서 우리는 출근을 하기 위해서 또는 등교를 위해서 등 아침 일찍 정해진 시간에 맞춰 일어나야만 한다. 잠에서 깨기 위해 일반적으로 휴대전화 앱 기능 중 하나인 알람을 사용한다. 일어나야 하는 시간에 알람을 맞춰, 설정한 시간에 휴대전화에서 알람음이 울리게 하거나 진동을 울리게 하여 잠에서 깰 수 있다.
그러나 알람을 듣고도 수면 시간이 적었거나 기본 수면 시간 많이 필요한 사람은 알람을 듣고도 일어나지 못할 가능성이 있다. 알람을 듣고 깼었지만 잠에서 깨지 못하고 다시 잠들거나, 잠에서 깨지 못했지만 습관적으로 알람을 끄고 계속 잠을 잔 경험이 있을 것이다. 이러한 경우에 휴대전화의 알람은 제 기능을 하지 못하고 무용지물이 되어버린다.

휴대전화 알람 기능은 보통 화면에 있는 알람 끄기 버튼을 누르거나 슬라이드 하는 식으로 알람을 끌 수 있다. 알람 해제 방법의 단순함 때문에, 앞서 말한 수면 중에도 습관적으로 알람을 끌 수 있는 문제가 발생한다고 생각했다. 또한 알람 해제 방식이 정적이지 않고 활동적이라면 잠에서 깨지 못하는 문제도 해결할 수 있을 것이다. 따라서 휴대전화 알람 기능 해제 방식의 단순함을 해결하기 위해 Jumplime을 고안하였다. 알람을 설정한 시간에 알람음 또는 진동이 울리고 Jumplime이 실행된다. 게임에서 일정 점수 이상을 획득하면 알람을 해제할 수 있다.

따라서 Jumplime은 평소 잠에서 깨는데 어려움이 있는 사람, 기존 휴대전화 알람의 효과를 보지 못했던 사람 등이 주 타겟이 될 것이다. 알람 해제를 위해 게임을 진행해야 하므로 잠에서 확실히 깰 수 있게 해줄 것이기 때문이다.

해당 문서는 Analysis에 이어 세 번째 단계인 Design에 관한 내용을 다룬다. 실제 구현에서 사용될 요소들을 확정하고 구체적으로 설계하는 단계이다. 모든 요소들을 명시하고 Diagram을 통해 전반적인 시스템의 흐름을 시각화하여 보여준다.

2. Class diagram



[그림 1] Jumplime의 Class diagram

1) GameStart

(1) Attributes

+ startImage:GameObject : 시작 이미지
+ gameStarted:bool : 게임 시작 여부

(2) Methods

- Start():void : 시작 이미지 출력
- Update():void : 마우스 왼쪽 클릭하면 게임 시작으로 판단
- StartGame():void : 시작 이미지 없앰

2) GameQuit

(1) Methods

- Update():void : ESC 키가 눌리면 게임 종료로 판단
- ExitGame():void : 게임 종료

3) CameraAction

(1) Attributes

+ target:GameObject : 플레이어
+ moveSpeed:float : 카메라 이동 속도
+ distanceY:float : 플레이어와 카메라의 Y축 거리
+ targetPos:Vector3 : 플레이어의 위치

(2) Methods

- Update():void : 플레이어의 위치로 카메라 위치 이동

4) PlayerMovement

(1) Attributes

+ maxPower:float : 최대 파워
+ powerIncreaseRate:float : 파워 증가 속도
+ speed:float : 이동 속도
+ power:float : 파워
+ minX:float : x의 최소값
+ maxX:float : x의 최대값
+ animator:Animator : 플레이어 애니메이션
+ rigidbody:Rigidbody2D : 플레이어 Rigidbody
+ isDown:bool : 클릭 여부
+ isJump:bool : 점프 여부
+ startPos:Vector3 : 시작 위치

(2) Methods

- Start():void : 플레이어 애니메이션과 플레이어 Rigidbody 정보를 가져옴, x의 최소값과 x의 최대값을 카메라 넓이로 계산하여 저장
- Update():void : 점프 상태이면 이동 불가
- LateUpdate():void : 플레이어 x축 이동 제한
- CheckMouseState():void : 마우스 클릭 이벤트 처리, 마우스 왼쪽 버튼을 누르면 파워 증가, 마우스 왼쪽 버튼을 떼면 마우스 방향으로 파워만큼 플레이어 이동
- Shoot(Vector3):void : 플레이어를 마우스 클릭을 뗀 방향으로 파워만큼 이동
- OnCollisionEnter2D(Collision2D):void : 플레이어가 타일과 접촉하면 점프 가능 상태 해제
- LimitToMove():void : 플레이어 x축을 x의 최소값에서 최대값 사이로 제한

5) PlayerController

(1) Methods

- Update():void : 플레이어의 y축 위치를 계산하여 높이 점수를 갱신

- OnTriggerEnter2D(Collider2D):void : 플레이어와 사과의 충돌 감지, 충돌한 사과가 사과이면 사과 점수 30 증가, 충돌한 사과가 벌레 먹은 사과이면 10 감소

6) ScoreManager

(1) Attributes

+ totalScore:int : 총 점수
+ heightScore:int : 높이 점수
+ appleScore:int : 사과 점수

(2) Methods

- Awake():void : 총 점수, 높이 점수, 사과 점수를 0으로 초기화
- Update():void : 높이 점수와 사과 점수의 합은 총 점수, 현재 총 점수를 계산하고 저장

7) ScoreUpdate

(1) Attributes

- scoreText:Text : 점수 텍스트

(2) Methods

- Awake():void : 점수 텍스트 정보를 가져옴
+ Update():void : 현재 총 점수를 화면에 출력함

8) Tile

(1) Attributes

+ maxDistance:float : 제거되지 않는 최대 거리
- collider:BoxCollider2D : 타일의 BoxCollider2D
- player:GameObject : 플레이어
- isStartTile:bool : 시작 타일 여부

(2) Methods

- Start():void : 타일의 BoxCollider2D와 플레이어 정보를 가져옴, 시작 타일 여부 판별
- Update():void : 플레이어와 최대 거리보다 멀어지면 해당 타일 제거
- OnTriggerEnter2D(Collider2D):void : 플레이어와의 충돌이 발생하면, 플레이어가 해당 타일보다 밑에 있을 경우 충돌 무시, 위에 있을 경우 충돌 적용
- OnTriggerStay2D(Collider2D):void : 플레이어와의 충돌이 발생하고 있으면, 플레이어가 해당 타일보다 밑에 있을 경우 충돌 무시, 위에 있을 경우 충돌 적용
- OnTriggerExit2D(Collider2D):void : 플레이어와의 충돌이 끝나면, 해당 타일의 충돌 적용

9) SpawnTile

(1) Attributes

+ player:GameObject : 플레이어
+ tilePrefab1:GameObject : 타일 종류1
+ tilePrefab2:GameObject : 타일 종류2
+ minX:float : x의 최소값
+ maxX:float : x의 최대값
+ minY:float : y의 최소값
+ maxY:float : y의 최대값
+ spawnDelay:float : 생성 딜레이
- tileWidth1:float : 타일 종류1의 너비
- tileWidth2:float : 타일 종류2의 너비
- lastSpawnPosY:float : 마지막 생성 타일의 y 위치
- newTilePrefab:GameObject : 새로운 타일의 종류
- newTileWidth:float : 새로운 타일의 너비

(2) Methods

- Start():void : 타일 종류1, 타일 종류2, 플레이어의 정보를 가져옴, 마지막 생성 타일의 y 위치를 플레이어의 y 위치로 저장
- Update():void : 시간이 생성 딜레이보다 크거나 같으면 새로운 타일 생성하고 생성 딜레이는 1에서 3의 수 중에서 랜덤하게 증가
+ SpawnNewTile():void : 타일 종류1과 타일 종류2 중 랜덤한 타일 종류를 골라 새로운 타일의 종류와 새로운 타일의 너비에 정보 저장, 생성할 x, y 위치를 각각의 최소값과 최대값 사이의 범위에서 랜덤하게 선정함, 해당 위치에 새로운 타일 생성, 마지막 생성 타일의 y 위치는 생성한 타일의 y 위치로 저장

10) SpawnApple

(1) Attributes

+ player:GameObject : 플레이어
+ applePrefab1:GameObject : 사과 종류1
+ applePrefab2:GameObject : 사과 종류2
+ minX:float : x의 최소값
+ maxX:float : x의 최대값
+ minY:float : y의 최소값
+ maxY:float : y의 최대값
+ spawnDelay:float : 생성 딜레이

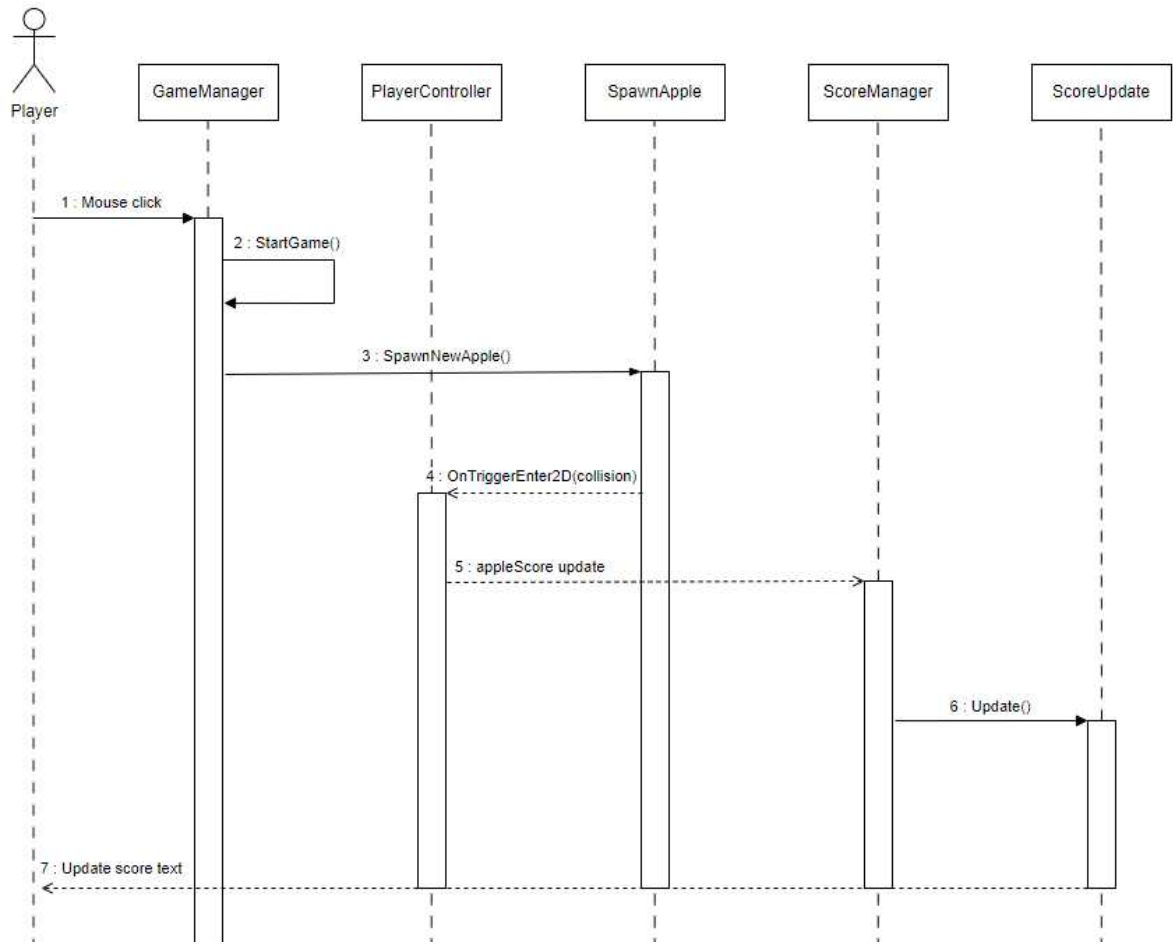
- lastSpawnPosY:float : 마지막 생성 사과의 y 위치
- newApplePrefab:GameObject : 새로운 사과의 종류

(2) Methods

- Start():void : 플레이어의 정보를 가져옴, 마지막 생성 사과의 y 위치를 플레이어의 y 위치로 저장
- Update():void : 시간이 생성 딜레이보다 크거나 같으면 새로운 타일 생성하고 생성 딜레이는 1에서 3의 수 중에서 랜덤하게 증가

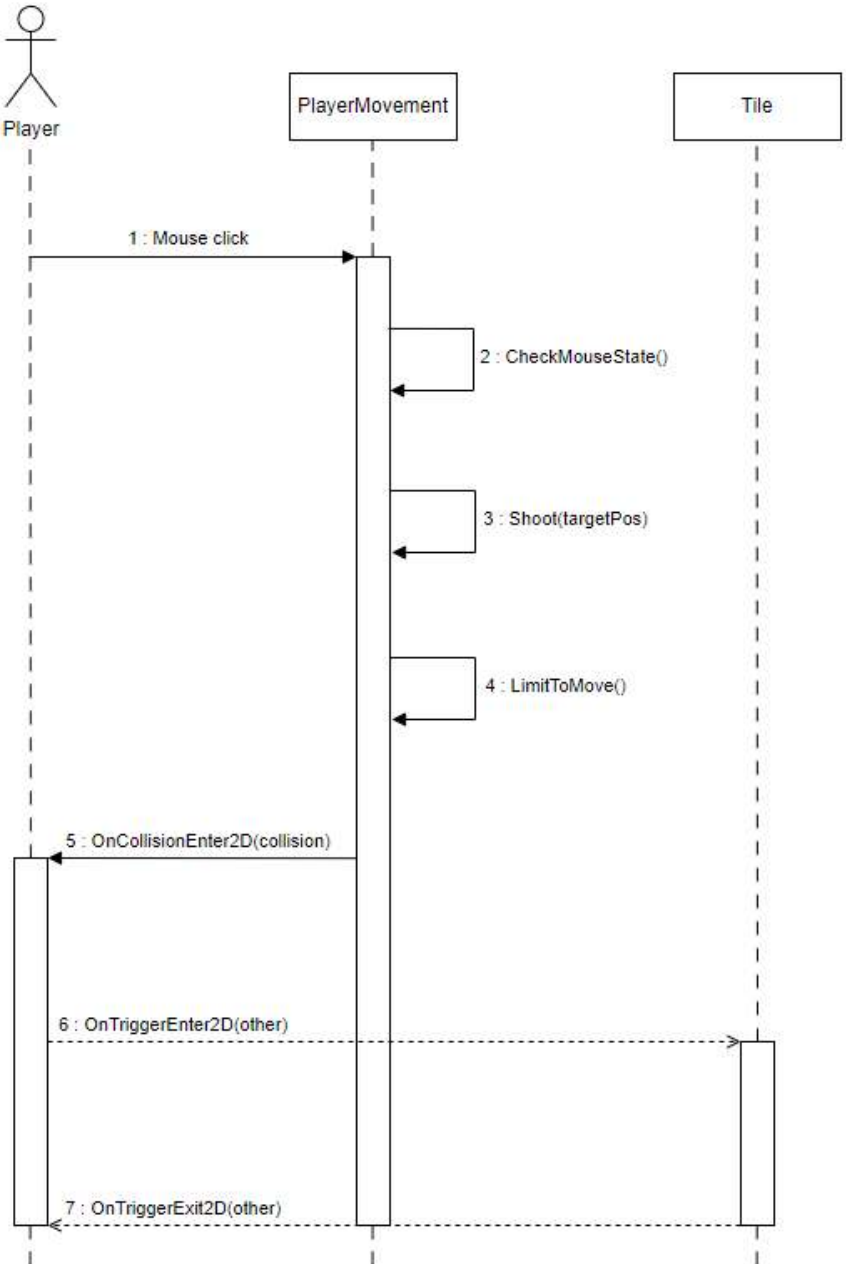
+ SpawnNewApple():void : 사과 종류1과 사과 종류2 중 랜덤한 사과 종류를 골라 새로운 사과의 종류에 정보 저장, 생성할 x, y 위치를 각각의 최소값과 최대값 사이의 범위에서 랜덤하게 선정함, 해당 위치에 새로운 타일 생성, 마지막 생성 타일의 y 위치는 생성한 타일의 y 위치로 저장

3. Sequence diagram



[그림 2] Score Sequence diagram

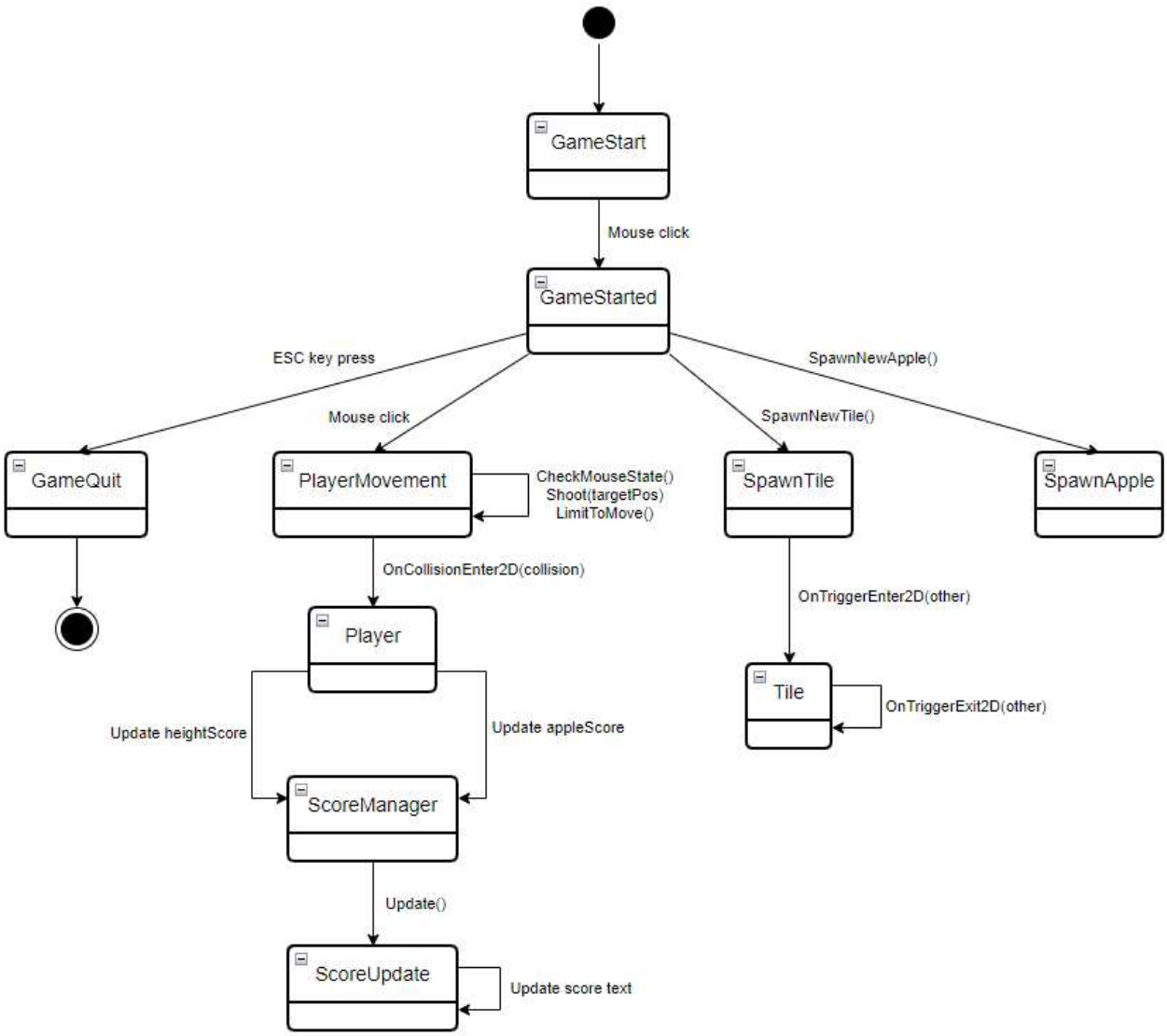
위 그림은 게임이 시작하고 플레이어의 이동을 통제하는 과정이다. 플레이어가 마우스 왼쪽 버튼을 누르면 GameManager가 게임을 시작한다. 그 후, 사과와 타일이 랜덤으로 생성되기 시작한다. 플레이어가 이동을 통해 사과와 접촉하면 사과 점수를 갱신한다. 변경된 총 점수를 계산하여 게임 화면 우측 상단에 출력한다.



[그림 3] Player Movement Sequence diagram

위 그림은 플레이어가 마우스에서 발생한 이벤트에 대해 이동하는 과정이다. 마우스 클릭 이벤트가 발생하면 발생한 마우스 이벤트의 상태를 확인하고 상태에 따른 동작을 실행한다. 마우스 클릭이 유지되는 상황에서, 즉 마우스 왼쪽 버튼을 누르고 있는 경우에는 플레이어의 이동 파워가 증가한다. 마우스 클릭을 뗀 경우, 그 방향으로 이동 파워 만큼 플레이어를 이동한다. 플레이어가 타일과 충돌하였을 때, 플레이어가 해당 타일보다 낮게 위치한 경우 타일의 충돌을 무시한다. 만약 플레이어가 해당 타일보다 높게 위치한 경우 타일에 충돌을 적용하여 플레이어가 타일 위로 올라갈 수 있게 한다.

4. State machine diagram



[그림 4] Jumplime의 State machine diagram

시스템을 실행하면 게임 시작 화면이 출력되며 마우스 왼쪽 클릭을 할 시, 게임 화면으로 이동한다. 게임이 시작되면 마우스 왼쪽 클릭을 통해 플레이어를 이동할 수 있다. 마우스 왼쪽 버튼을 누른 시간만큼 플레이어의 이동 파워가 증가하고 마우스 왼쪽 버튼을 뗀 방향으로 이동 파워 만큼 플레이어가 이동한다. 이동하면서 플레이어의 높이에 따라 높이 점수를 갱신하고 플레이어와 접촉한 사과의 종류에 따라 사과 점수를 증가하거나 감소하여 계산 후 사과 점수를 갱신한다. 또한 갱신된 점수를 게임 화면 우측 상단에 표시하여 사용자가 점수를 실시간으로 확인할 수 있게 한다. 게임 화면에서 타일과 사과는 랜덤 위치에 생성된다. 게임 화면에서 ESC 키를 누르면 게임이 종료된다.

5. Implementation requirements

1) H/W platform requirements
   (1) CPU : Intel PENTIUM IV 이상
   (2) RAM : 1GB 이상
   (3) HDD/SSD : 10GB 이상

2) S/W platform requirements
   (1) OS : Microsoft Window 7 이상
   (2) Implemetaion Language : C#

6. Glossary

Terms	Description
Class diagram	객체 지향의 설계 단위인 Class를 이용하여 시스템의 정적인 구조를 표현한 다이어그램이다.
Sequence diagram	객체 사이의 동적인 상호작용을 시간적 개념을 기준으로 모델링한 다이어그램이다.
State Machine diagram	객체의 lifetime 동안 변환될 수 있는 모든 상태를 정의해둔 다이어그램이다.

7. References

1) Class 다이어그램 제작 참고:
https://support.microsoft.com/ko-kr/office/uml-%ED%81%B4%EB%9E%98%EC%8A%A4-%EB%8B%A4%EC%9D%B4%EC%96%B4%EA%B7%B8%EB%9E%A8-%EB%A7%8C%EB%93%A4%EA%B8%B0-de6be927-8a7b-4a79-ae63-90da8f1a8a6b

2) Sequence 다이어그램 제작 참고:
https://support.microsoft.com/ko-kr/office/uml-%EC%8B%9C%ED%80%80%EC%8A%A4-%EB%8B%A4%EC%9D%B4%EC%96%B4%EA%B7%B8%EB%9E%A8-%EB%A7%8C%EB%93%A4%EA%B8%B0-c61c371b-b150-4958-b128-902000133b26