|  |
| --- |
| AD프로젝트 보고서  2019155김규민 20191566김유빈 |
|  |

# 프로젝트 소개

|  |
| --- |
| 저희 프로젝트는 포레스트 템플이라는 게임에 착안하여서 그와 유사하게 게임을 제작하였습니다. 그렇기 때문에 실제 포레스트 템플과 비슷하게 버튼을 밟으면 어떤 오브젝트가 반응을 하던가 캐릭터와 다른 색의 물을 밟으면 캐릭터가 죽게 되도록 프로젝트 하였습니다. |

# 코드

|  |
| --- |
| PLayer.py \_\_init\_\_  변수들의 값 설정  key\_\_in  if self.x()를 통해서 현재 캐릭터가 화면을 벗어났는지 인식하여 true, false 리턴  if end\_R, end\_L 에서는 if self.x() 에서 리턴된 값이 True 일 경우 화면을 벗아날수없게 캐릭터의 가속도를 0으로 변환  if Qt.Key\_Left, Right 에서는 만약 움직임 키가 눌리고 있고 if self.x()에서 false 가 리턴됬다면 캐릭터가 움직일수 있도록 가속도 부여  if Qt.Key\_Up (?)  gravity  if self.jumped 를 통해 캐릭터가 점프를 한 상황이 인식되면 수직가속도(중력)의 크기를 증가  ground\_detect  self.foot\_y를 통해 캐릭터 발의 높이를 따로 설정  (?)  Jump  if self.jumped 를 통해 캐릭터가 점프(?)  if self.y\_before\_jump를 통해 점프전 발의 위치와 현제 발의 위치를 뺸값이 최대 점프높이에 도달하면 점프했다고 인식하면 jumped 값을 false 로 리턴  move\_per\_frame  if self.excel\_horizontal 에서 좌,우 가속도가 최대 가속도가 되었을 때 값이 넘어가지 않고 최대 가속도를 유지할수 있도록 설계  if self.standing 에서 움직임이 없다고 인식되면 좌,우 가속도를 0으로 설정  self.setPos(?)  inertia  (?)  char\_animate (설명 부가?)  if self.excel\_horizontal을 통해 가속도의 방향을 구해서 그 방향에 맞는 캐릭터의 프레임을 반복적으로 구현하였고 excel\_horizontal 값이 0이 나오면 캐릭터의 모습을 가만히 서있는 그림으로 구현(?) |
| map\_object.pyclass f\_door, w\_door\_\_init\_\_을 통해서 문의 이미지를 “f,w\_door\_0.png”로 설정하고 문의 크기를 조절초반에는 문이 열려있지 않도록 opened 값을 false 로 설정open 에서는 collidWithItem 의 값이 True 일떄 각각의 문이 열리도록 이미지를 “door\_opened.png”로 변환하고 opended 값을 false 로 변환하여서 문이 열린 것을 인식class SolidRect발판오브젝트의 크기, 기준위치, 색 설정class PoolFire, PoolWater\_\_init\_\_을 통해서 각각 오브젝트에 이미지를 설정, setPos를 통해서 오브젝트의 위치기준 생성, setScale을 통해서 (?)kill\_water, kill\_fire 을 통해서 각각의 colidesWithItem()이 True 가 될 때 water.set Visible(False), fire.setVisible(False)을 통해서 캐릭터가 죽도록 설정class PoolPoison\_\_init\_\_은 위 Poolfire, Poolwater 의 \_\_init\_\_ 과 동일kill\_all 에서는 캐릭터가 이 오브젝트에 닿을시 모두 죽도록 설정class Button\_\_init\_\_에선 버튼의 기준 위치, 버튼의 색을 설정하고 pushed = False 을 통해서 버튼이 눌리지 않았음을 인식, self.slide = slide(?)push\_tect에서는 플레이어 1,2 의 colidesWithItem이 True가 됬을 때 pushed 값에 True를 넣어서 버튼이 밟혔음을 인식하고 다시 버튼을 밟지 않는 상황이 되면 pushed 값에 false 를 넣어서 버튼이 밟히지 않음을 인식class Slide\_V(버튼 인식을 통한 슬라이더 움직임?)\_\_init\_\_을 통해서 |
| scene.py class Title(?)    class Scene0, Scene1, Scene2  앞서 만들었던 오브젝트와 캐릭터를 이용해 게임 내에서 스테이지 구현하고 캐릭터와 오브젝트의 상호작용 구현  class end(?) |
| Main\_screen.pyClass MainWindow(?)QGraphicsScene으로 윈도우 크기 설정하고 setBackgroundBrush를 통해 백그라운드 색을 검정으로 설정class changScene각각의 캐릭터가 적절한 문에 도달했을 때 스테이지가 클리어 된 것을 인식하여 다음 스테이지로 넘어가도록 설계 |

# 발견된 오류

|  |  |
| --- | --- |
| * 오류1 * 오류2 * 오류3 * 오류4 |  |

# 수정 방향