**2021년도 프로그래밍방법론및실습**

**프로젝트 최종보고서**

**[Getout]**

제출일자: 2021. 12. 21

팀원1: 2018430006 김도형

팀원2: 2018430013 김영준

**- 목차 -**

1. 프로젝트 개요

1. 프로젝트 소개
2. 개발 범위 및 내용
3. 개발 목표

2. 세부 개발 내용

1. 개발 과정 및 추진 체계
2. 빌드 환경 및 플랫폼
3. 소스코드 구조
4. 주요 알고리즘 및 구현 소개

3. 동작 설명

1. 프로그램 동작 화면
2. 개발 목표 달성도

4. 결론

**1. 프로젝트 개요**

**가. 프로젝트 소개**

[SDL2를 활용하여 C언어로 프로그래밍]

SDL2 라이브러리를 이용한 Top-down 형식의 게임을 구현하였다. 게임을 대표할 수 있는 키워드는 공포, 방탈출, 액션, 퍼즐이다. 우선 게임속의 배경에 대해 정리하자면, 주인공(player)이 납치범에게 납치되었고, 납치범 자택의 좁은 방 안에서 깨어나는 것부터 게임이 시작된다. 주인공은 첫번째 방 안의 오브젝트(퍼즐, 힌트)들을 활용하여 잠겨 있는 방문을 열고 두번째 방으로 나가야한다. 이때 해결해야하는 퍼즐로는 전선회로문제가 난이도별로 3가지가 있고, 키 패드 자물쇠를 풀기 위한 힌트도 주인공이 방 안 곳곳을 수색하며 찾아야 한다.

 두번째 방인 넓은 거실에서는 납치범이 무기를 들고 주인공을 쫒아오기 시작한다. 주인공은 위험해 보이는 오브젝트들을 피하면서 무기(샷건)을 찾아 범죄자를 죽이고 현관문으로 집을 탈출해야 한다.

최소한의 가이드와 힌트를 통해 플레이어가 자신의 추리력을 최대한 활용할 수 있도록 유도하였고, 의도적인 게임 시스템 배치에서 나오는 어색함을 없앴다.

무한히 이어지는 게임이 아닌 스토리와 엔딩을 가지는 게임 제작을 희망해 플레이어와 플레이어 주변의 상호작용을 중점으로 생각한 장르를 설정하였다. 이를 위해 프로젝트 주제로는 플레이어 주변 사물들을 활용하여 방을 탈출하는 게임 장르인 방탈출게임을 참고하였으며, 방을 탈출하기 위한 퍼즐로는 1학년 교과목 디지털 논리설계의 게이트에 따른 신호 변환, 게임 ‘어몽어스’의 비밀번호 시스템, 방탈출 게임에서 자주 보이는 주변 사물을 통한 패스워드 도출 등을 이용하였다. 퍼즐을 풀어 방을 탈출한 뒤에는 샷건을 이용한 액션을 추가하여 기존 방탈출 게임에서 느낄 수 있는 게임 플레이의 단조로움을 없앴고, 기존 슈팅게임과는 다른 독창적인 플레이 방식을 만들기 위해 새로운 방식을 추가하기로 하였다.

 넷플릭스 “DP”에서 사람을 벽으로 밀쳐서 벽에 박힌 못에 의해 뒤통수에 해를 가하는 장면을 보고 사람을 밀쳐서 간접적으로 상대에게 피해를 줄 수 있다는 아이디어를 얻었다. 게임 속 주인공은 샷건으로 납치범을 쏴보지만 납치범의 체력은 전혀 깎이지 않고 밀려나기만 한다는 것을 알게 되고, 이때부터 주인공은 주변물(벽에 박힌 못, 벽난로 등등)을 활용해 납치범을 죽여야 한다는 생각을 얻게 된다.

**나. 개발 범위 및 내용**

SDL 라이브러리와 C 언어를 이용, 2개의 맵과 3개의 퍼즐, 액션을 포함하여 7분정도의 플레이 타임을 가진, 플레이어의 명확한 목적, 스토리와 엔딩이 존재하는 게임 개발한다.

‘디지털논리회로및실습’ 수업 때 배운 디지털 신호 및 게이트를 활용하기 때문에 교육적인 측면이 있으며, 또한 기존에 없던 주변물과의 상호작용을 통한 액션게임 경험을 통해 플레이어에게 색다른 재미를 선사한다.

**다. 개발 목표**

|  |  |  |
| --- | --- | --- |
| 평가 항목 | 개발 목표 | 우선 순위 |
| 컴파일 및 링킹 | 수천줄의 소스코드를 목적성에 맞게 헤더파일을 나누어 가독성과 효율성이 높도록 분배하였다. | 1 |
| 디버깅 및 최적화 | 빌드 및 플레이에 있어 오류가 없어야 하며, 플레이어가 취할 수 있는 행동을 예측하여 예상치 못한 일이 일어나는 일을 방지한다. | 1 |
| 난이도 | 플레이어가 주변 상황을 인식하여 막힘없이 플레이 하되, 너무 쉽다고 느껴지지 않을 정도로 난이도를 조절한다. | 2 |
| 창의성 | 특징이 각각 다른 오브젝트들을 배치하여 플레이어가 다양한 경험을 할 수 있도록 한다. | 2 |
| 편의성 | 게임 조작에 있어 플레이어가 불편함을 느끼지 않도록 편의성을 고려한 코드를 작성한다. | 3 |

**2. 세부 개발 내용**

**가. 개발 과정 및 추진 체계**

팀원별 역할 분담

|  |  |
| --- | --- |
| 팀원 | 역할 |
| 김도형 | 첫번째 방의 논리회로 퍼즐, 패스워드 퍼즐 등의 퍼즐 시스템 구성,  게임플레이 힌트 팝업, 맵 이동 및 키보드, 마우스를 이용한 상호작용 코드 구성 |
| 김영준 | 두번째 방에서 주인공과 납치범의 상호작용 및 방에 배치되는 오브젝트들(주인공을 해치는 주변물, 납치범을 해치는 주변물, 무기) 설정, 오브젝트 및 맵 이미지 디자인 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 세부 개발 내용 | 담당자 | 11/1주 | 11/2주 | 11/3주 | 11/4주 | 12/1주 | 12/2주 | 12/3주 |
| 게임 컨셉 +아이디어 구상 및 역할분담 | 팀 |  |  |  |  |  |  |  |
| 코드 디버깅 및 최적화 | 팀 |  |  |  |  |  |  |  |
| 마우스 이용 상호작용 구현 및 논리회로 퍼즐 구성 | 김도형 |  |  |  |  |  |  |  |
| 키보드 이용 상호작용 구현 및 힌트 팝업, 맵 이동 코드 구성 | 김도형 |  |  |  |  |  |  |  |
| 패스워드 퍼즐 구성 | 김도형 |  |  |  |  |  |  |  |
| 주인공과 납치범 이동방식 구성, 이미지 디자인 | 김영준 |  |  |  |  |  |  |  |
| 샷건 작동 방식, 발사 방향 설정 | 김영준 |  |  |  |  |  |  |  |
| 오브젝트 상호작용 구성, 이미지 디자인 | 김영준 |  |  |  |  |  |  |  |
| 최적화 | 팀 |  |  |  |  |  |  |  |

**나. 빌드 환경 및 플랫폼**

|  |  |
| --- | --- |
| 항목 | 사양 |
| 운영 체제 | Virtualbox Ubuntu 18.04 |
| 개발 언어 | C |
| 컴파일러 | gcc 5.4.0 |
| 사용 라이브러리 | SDL2 version2.0.4, make 4.1, math.h, time.h |

**빌드 방법**

텍스트이(가) 표시된 사진

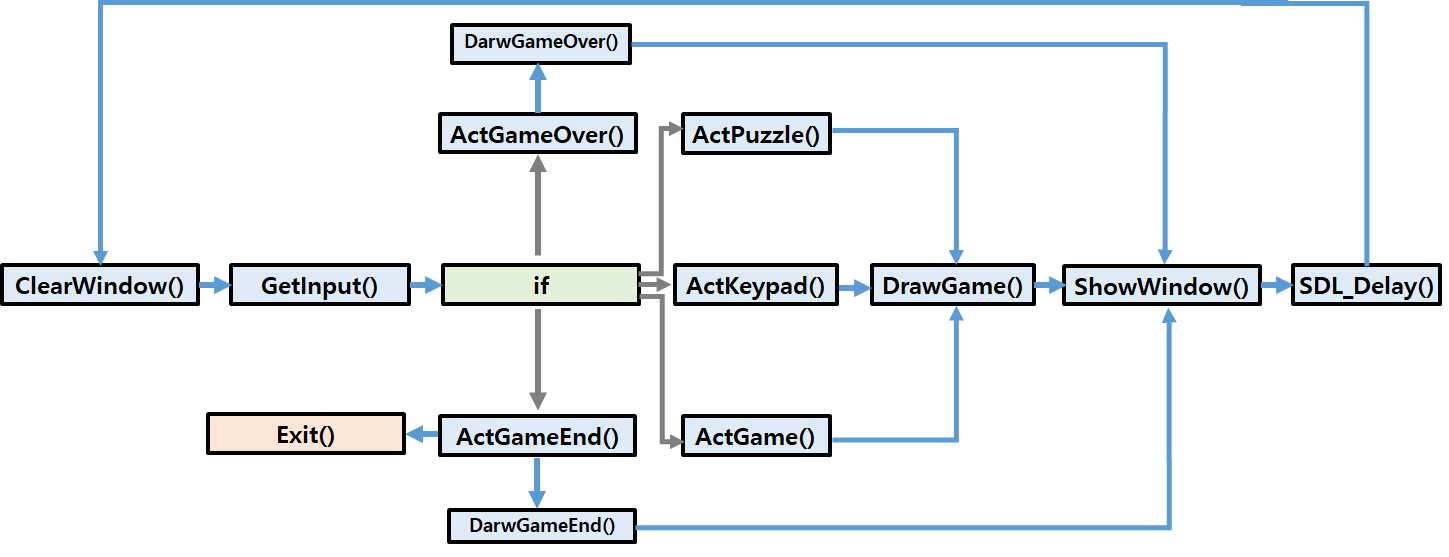
자동 생성된 설명

Getout 파일 내의 src 파일으로 디렉터리를 변경한 후, makefile을 통해 src내의 모든 소스파일을 일괄적으로 컴파일한다. 그리하면 상위 디렉터리에 실행파일이 생성된다.

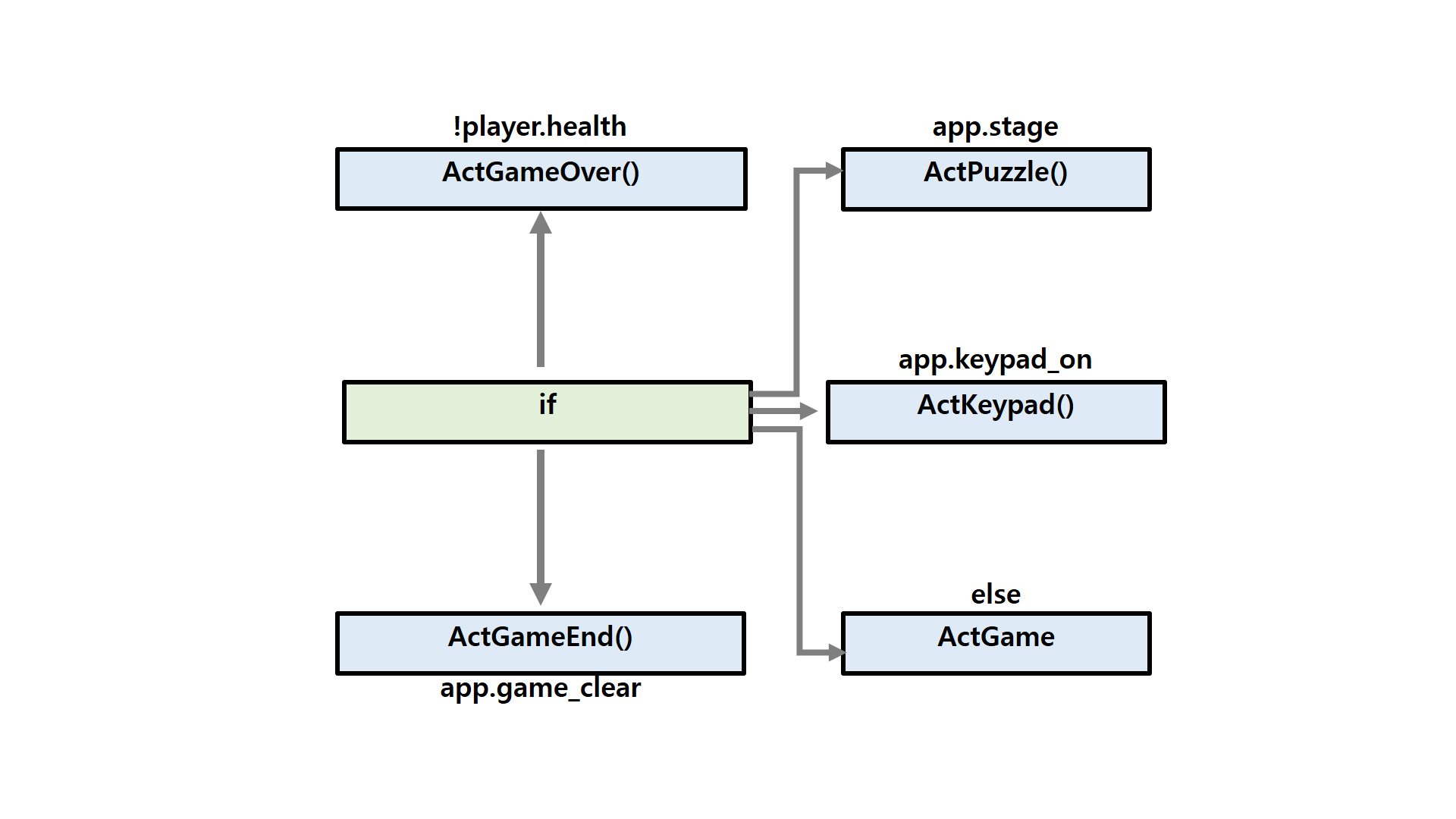
|  |  |  |
| --- | --- | --- |
| 폴더/파일 | | 설명 |
| bin | | 컴파일 이후 실행 파일이 저장될 폴더 |
| gfx | | 오브젝트, 맵, 퍼즐 등의 이미지파일이 저장된 폴더 |
| src | main.c | 게임 플레이 main loop를 구현한 파일 |
| main.h | main.c의 정의가 담긴 헤더 파일 |
| init.c | 각종 오브젝트와 struct를 초기화하는 함수를 구현한 파일 |
| init.h | init.c에 대한 헤더 파일 |
| input.c | 키보드/마우스 입출력 관련 함수를 구현한 파일 |
| input.h | input.c에 대한 헤더 파일 |
| draw.c | 그래픽 렌더링 관련 함수를 구현한 파일 |
| draw.h | draw.c에 대한 헤더 파일 |
| utils.c | 기타 부가적으로 활용되거나 개발 시의 편의성을 위해 활용되는 함수들을 선언한 파일 |
| utils.h | 기타 필요한 구조체 및 열거형 변수들을 선언한 헤더 파일 |
| makefile | 프로젝트 컴파일 환경을 저장하고 있는 makefile |

**라. 주요 알고리즘 및 구현 소개**

**main함수 루프 구조**



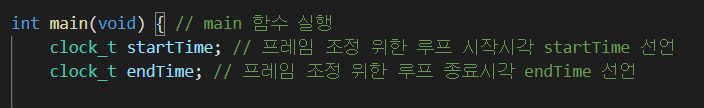
ClearWindow()함수를 통해 화면을 초기화하고, GetInput()함수를 통해 입력을 받은 뒤, if 조건문을 통해 화면에 실행할 Act 함수를 결정한다. 실행 함수 결정 방법은 다음과 같다.



플레이어 체력이 0이면 ActGameOver()함수, 게임 클리어 시에는 ActGameEnd()함수가 실행되고, 논리 회로 퍼즐 실행 시 ActPuzzle()함수, 키패드 퍼즐 실행 시 ActKeypad()함수, 이외에는 ActGame()함수가 실행된다.

이후 Drawgame()함수를 통해 화면을 렌더링하고, ShowWindow()함수를 통해 화면을 출력한다.

그 후 루프 실행에 걸린 시간을 통해 딜레이를 계산해 SDL\_Delay() 함수를 통해 프레임에 따라 딜레이를 주고 루프의 처음으로 돌아온다.



main() 함수에서 선언된 clock\_t 타입의 startTime과 endTime에 clock()함수를 통해 루프 시작 시점과 종료 시점을 할당하게 된다

텍스트이(가) 표시된 사진

자동 생성된 설명

이후 루프 종료 시점에서 루프 시작 시점을 빼 루프 실행에 걸린 시간을 구하고, 한 번의 루프에 16ms의 시간이 걸리도록 SDL\_Delay()함수를 통해 딜레이를 조절한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

플레이어와 물체의 상호작용을 구현하기 위해 투명한 object인 interact\_obj[]를 만들어 CheckCollisionObjects() 함수를 통해 오브젝트의 인덱스에 따른 상호작용을 간편하게 구현하였다.

텍스트이(가) 표시된 사진

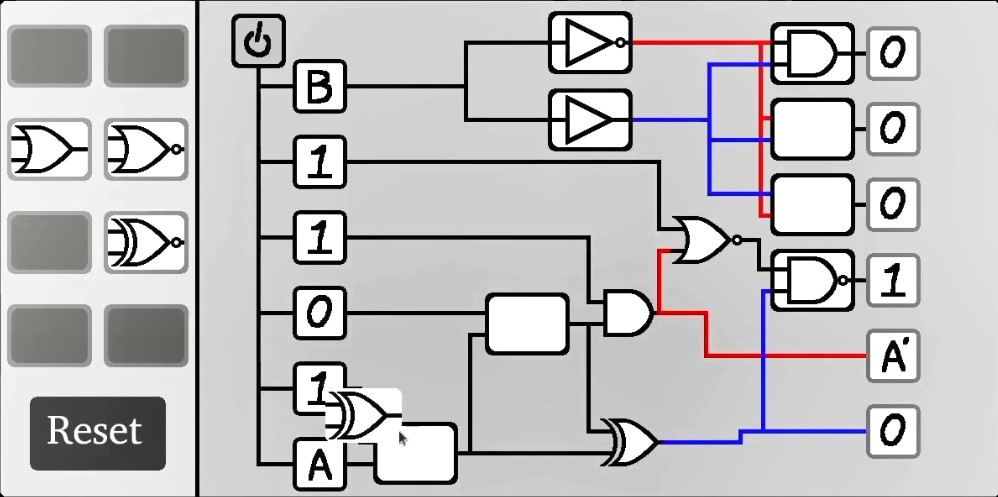
자동 생성된 설명

구조체와 퍼즐에 대한 오브젝트 Clear 함수를 생성하여 코드에 간결성을 더하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

논리 검사 함수를 생성해 어떠한 입력 신호나 게이트 조합에도 올바른 출력 신호를 도출해 낼 수 있도록 코드를 설계하였다.



ActPuzzle() 함수 실행 시 논리게이트 칩을 옮겨 올바른 출력을 완성하는 논리 게이트 퍼즐이 실행된다. 좌측에 있는 게이트 칩을 옮긴 후 전원 버튼을 눌러 올바른 신호가 출력되면 퍼즐이 종료된다. 논리 게이트 칩을 옮기는 과정에서 게이트가 슬롯에 가까이 가면 자동적으로 게이트를 배치하고, 게이트를 슬롯 밖에 놓았을 때 게이트의 원래 위치에 들어가고, 마우스 우클릭을 통해 게이트 이동을 취소하였을 때 원래 있던 슬롯의 위치를 기억해 돌아오는 여러 편의성 코드가 포함되어 있다.

이후 전원버튼을 눌렀을 시 ActPower() 함수가 실행되어 슬롯에 있는 게이트 종류를 확인하고, Logic() 함수를 통해 출력 신호를 도출하고 Receiver의 신호와 비교하여 그 결과를 출력한다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명 ActKeypad() 함수 실행 시 마우스로 번호 키를 클릭해 키패드에 비밀번호를 입력하는 키패드 퍼즐이 실행된다. ActKeypad() 내부의 ActKeypress() 함수를 통해 app.passcode[] 멤버에 입력받은 숫자를 할당하고, 키패드 이미지를 교체한다. X버튼이 입력되면 입력 코드를 초기화하고, 체크버튼이 입력되면 PASSCODE 전역변수값과 비교하여 비밀번호가 일치하면 다음 map으로 이동한다.

**오브젝트 설명**

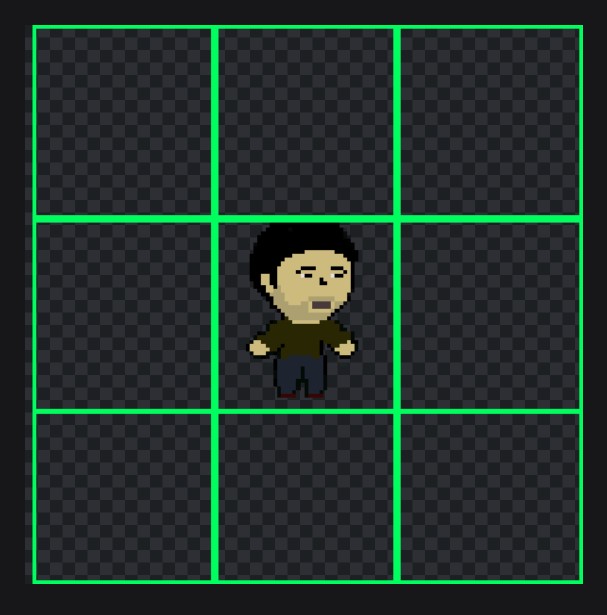
1) Shotgun 구현 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Shotgun의 픽셀크기는 player의 가로세로(48x48) 픽셀값의 3배값(144x144)을 할당하여 주었다. 처음 player가 샷건과 접촉하여 샷건을 소지한 상태가 되었을 때, 기본 장착상태는 ⑤번에 위치시켜 player가 샷건을 양손으로 파지하고 있는 이미지로 렌더링하게 된다.

Player가 spacebar를 눌러 enemy를 향해 샷건을 쏘는 상황에서, player와 enemy 사이의 상대적인 각도를 계산하여 ⑤번을 제외한 ①~⑨번 구간에 따라 샷건의 방향을 틀어 발사하는 이미지를 렌더링하도록 구현하였다. 예를 들어 enemy가 player로부터 11시 방향에 위치하였을 때, 샷건은 ①번 구간에서 발사되는 이미지를 렌더링하게 된다.



**①**

**②**

**③**

**⑥**

**⑤**

**④**

**⑦**

**⑧**

**⑨**

2) 다양한 오브젝트 소개

|  |  |  |
| --- | --- | --- |
| 오브젝트 이미지 | 이름 | 기능 |
|  | Trap (덫) | Player 또는 enemy가 덫을 밟으면 체력 1을 소모한다. 한번 사용된 후에는 접히게 되어 효력을 잃는다. |
|  | Fire (벽난로) | Player 또는 enemy가 불에 닿게 되면 체력 1을 소모한다. 한번 사용된 후에는 불이 꺼지게 된다. enemy에게 닿게 되면, enemy 객체는 이후에 불타는 이미지로 변경되며 속도가 줄어든다. |
|  | Screw (대못) | Player 또는 enemy가 대못에 박히게 되면 체력 1을 소모한다. 한번 사용된 후에는 사라져 효력을 잃는다. |
| 텍스트, 시계, 어두운이(가) 표시된 사진  자동 생성된 설명텍스트이(가) 표시된 사진  자동 생성된 설명 | Cactus(선인장) | Player 또는 enemy가 선인장과 부딪히게 되면 체력 1을 소모한다. 부딪힌 후에는 선인장이 망가져 효력을 잃게 된다. 만약 enemy에게 닿게 되면, enemy 객체의 속도는 상승하게 된다. |
|  | Sticky floor (끈적이는 바닥) | Player가 끈적이는 바닥을 지나는 동안 player의 속도를 줄이고 벗어나면 정상속도로 회복하게 된다. Enemy 객체에는 적용되지 않는다. |

3) Spider 객체 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

두번째 방에서 spawn되는 spider 객체는 player가 다가가게 되면 움직이기 시작하여 방 안을 돌아다니게 된다. 만약 player와 닿게 된다면 player는 사망하게 되므로 player는 spider를 유심히 주시하여 피해야 하는 게임 플레이방식이 추가되었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Spider는 벽에 닿게 되면 방향을 꺾어 계속 돌아다니도록 설계하였다. Spider 객체에 x\_val, y\_val 멤버변수를 할당하여 양옆 벽에 부딪치면 x\_val의 부호를 변경시켜 좌우 방향을 변경시키고, 위아래 벽에 부딪치면 상하 방향을 변경시켜 지속적으로 방 안에서 움직이도록 한다. 또한 RandInt()함수를 사용하여 spider객체를 연속적으로 미끄러지는 움직임이 아닌 (실제 벌레의 움직임과 유사하게) 불규칙적으로 움직이도록 구현하였다.

**3. 동작 설명**

**가. 프로그램 동작 화면**



첫번째 방에서 W,A,S,D 방향키로 player를 이동시키고 spacebar를 통해 방 안의 여러 오브젝트들을 확인하고 상호작용한다.

텍스트, 전자기기, 디스플레이이(가) 표시된 사진

자동 생성된 설명텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

마우스 드래그 드랍을 이용하여 논리 게이트 칩을 배치하고 올바른 출력신호를 완성시켜 퍼즐을 풀어 키패드의 전원을 복구한다. 키패드에서는 주어진 힌트를 활용하여 비밀번호를 알아내고 마우스를 이용해 키패드를 눌러 비밀번호를 맞춰 잠겨진 문을 열어 다음 방으로 이동한다.

텍스트, 벽, 실내, 여러개이(가) 표시된 사진

자동 생성된 설명

두번째 방에서 W,A,S,D 방향키로 player를 이동시켜 적과 spider를 피하면서 샷건을 찾아 획득하고 spacebar를 눌러 적을 밀쳐내는 액션게임을 진행한다. 또한 주변물들을 통해 적의 체력을 깎아 처치하는 미션을 수행한다.

]

**나. 개발 목표 달성도**

|  |  |  |
| --- | --- | --- |
| 평가 항목 | 개발 목표 대비 달성도 | 우선 순위 |
| 컴파일 및 링킹 | 10/10 | 1 |
| 디버깅 및 최적화 | 10/10 | 1 |
| 난이도 | 9/10 | 2 |
| 창의성 | 9.5/10 | 2 |
| 편의성 | 10/10 | 3 |

**4. 결론**

 SDL2 라이브러리를 활용하여 프로그램을 작성하며 라이브러리 활용 능력에 따라 코드 작성의 편의성과 개발 시의 자유도가 높아짐을 느껴 프로그램 작성 전 라이브러리 정보 수집의 중요성을 알 수 있었다. 또한, 리눅스 환경에서 각종 프로그램이나 게임의 기본적인 동작 원리를 알 수 있었으며, 디버그 과정을 통해 프로그램 실행 시 overhead를 최소화하고 예상치 못한 에러에 대응하기 위한 프로그래밍 기법을 체험할 수 있었다.  또한, 헤더파일과 makefile 파일을 이용하여 여러 코드를 동시에 컴파일하는 과정을 이해하는 데 도움이 되었다. 마지막으로, 현직 개발자가 팀원과 어떻게 협업하여 코드를 작성하는 지에 대한 실마리를 얻을 수 있었다.

 또한 프로그래밍 팀프로젝트는 처음 경험해 보았기 때문에 각자 소스코드를 작성하고 전체 코드를 통합하는 과정이 반복되는 작업에 있어 번거로운 부분이 있었다.  11월초에 git을 이용해 버전관리와 두 명이 동시에 작업이 가능하도록 분업 작업을 할 계획이었다. 따라서 gui로 쉽게 git 툴을 활용할 수 있는 Sourcetree라는 프로그램을 사용하려 했었다. 그러나 리눅스 환경에서 Sourcetree가 지원되지 않았고 터미널에 명령어를 통한 git 활용은 숙지가 안되어 있었으므로 이메일을 통해 각자 작성한 소스코드를 전달하고 각자 코드를 병합하는 과정을 반복하였다. 예상대로 번거로운 작업이 늘어났고 협업하는 과정에서의 아쉬움이 남았다. 그렇지만 git의 필요성에 대해 몸소 체험하게 되었고 이메일과 줌회의와 같이 온라인만으로 팀원과 협업하여 만족스러운 결과물을 도출해 냈기 때문에 뜻깊은 경험을 한 것 같다.