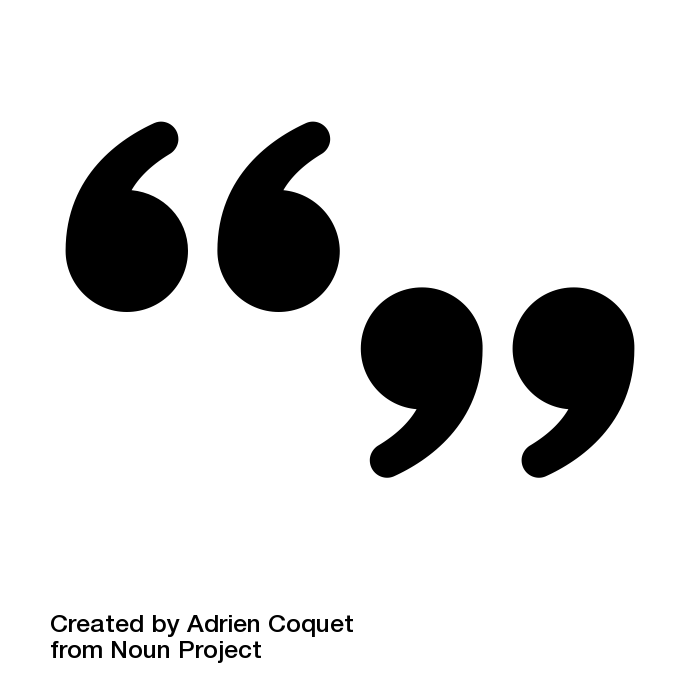
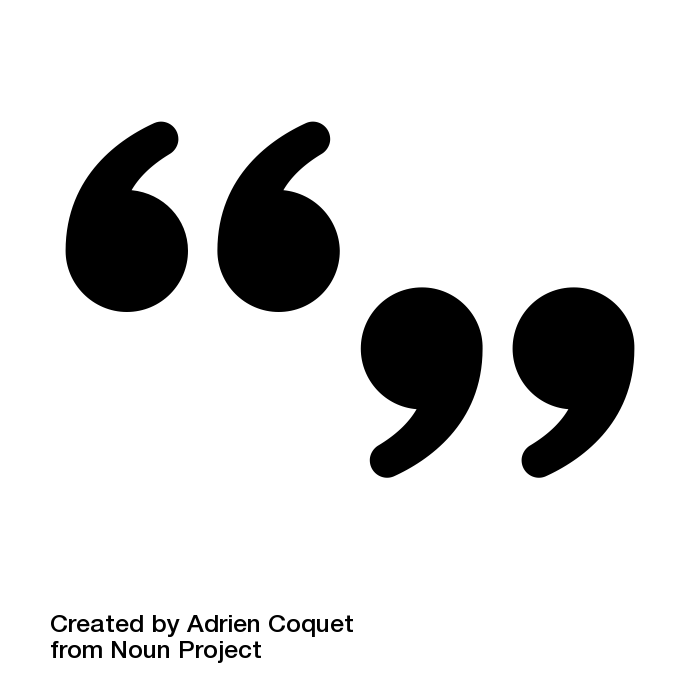
**파이썬 코딩을 통한 미니게임 패키지 프로그램 제작**

심서현 (영어영문학과)

본 프로젝트는 파이썬의 기본적인 문법의 활용을 통해 2000년대 초반 유행하였던 플래시 게임을 구현해보고, 다수의 게임 프로그램을 최대한 자연스럽게 구동할 수 있는 프로그램의 개발을 목표로 한다. 프로젝트의 개발은 파이게임을 기반으로 하는 파이썬 코딩을 통해 진행하였다. 프로그램의 구조는 다수의 게임을 포함 및 실행시킬 수 있는 게임 틀 프로그램과 3개의 게임 프로그램으로 구성되어 있다. 게임 틀 프로그램의 경우, 각 게임 코드와 변수 간 충돌 없이 게임이 실행되야 함을 고려하여 다수의 함수와 함수 내 변수를 통해 제작하였으며, 각 게임의 종료, 레벨 간 자연스러운 전환과 가시성을 위해 별도의 함수와 버튼 클래스를 제작하여 삽입하였다. 프로젝트에 사용한 게임으로는 공부수기 게임, 슈팅 게임, 테트리스 게임을 선정하였다. 공부수기 게임은 스프라이트의 이동 동선 설정 및 충돌 처리를 이용하여 게임 내 기믹을 구현하였다. 슈팅 게임은 공부수기 게임의 코드를 변형하고 클래스를 통해 다수의 객체를 처리할 수 있는 기능을 구현하였다. 테트리스 게임의 경우, 클래스 객체의 상태 갱신을 활용해 복합적인 상태 처리 기능을 구현하여 게임의 기믹을 제작하였다. 각 함수 내 변수의 변형 및 기믹의 추가를 통해 게임 별, 레벨 별 차별성을 두었으며, 게임의 단순화 방지와 편의성을 고려한 기능을 추가하였다. 결과적으로, 3개의 게임을 포함 및 실행할 수 있는 패키지 형태의 게임 프로그램을 구현하는데 성공하였으며, 파이썬 문법을 활용하여 코드의 결과를 가시화할 수 있는 프로그램을 제작할 수 있었다.

**1. 서론**

2020년 어도비의 플래시 프로그램이 서비스를 종료하기 전까지, 애니메이션과 게임 등 다양한 분야에서 플래시를 이용한 프로그램이 사용되었다. 특히 플래시 게임의 경우, 직관적인 프로그램 구조와 간편한 개발 구조를 가지고 있었기에, 다양한 기능 가진 게임을 빠르고 간편하게 제작할 수 있었다. 파이썬 코드 활용 숙지를 위해 다양한 파이썬 관련 서적과 강의를 통해 파이썬 활용에 대한 아이디어와 사례를 연구해 보던 도중, 플래시 게임의 특징과 변수 및 클래스 활용을 통해 프로그램을 간편하게 제작할 수 있는 파이썬의 특성 및 구조에서의 유사점을 느끼게 되었다. 그리하여, 한 때 플래시 게임으로 개발되었던 게임들을 파이썬을 통해 구현해 보자는 아이디어에서 시작하여, 파이썬 게임 프로그램 제작을 프로젝트 주제로 선정하게 되었다. 이 프로젝트는 파이썬의 기본적인 문법 및 개체 활용을 통해 플래시 게임 형태의 다수의 프로그램을 구현하고 최대한 자연스럽게 구동하는 프로그램을 개발하는 것을 목표로 한다. 더불어, 다수의 게임이 하나의 프로그램에 들어가 있는, 소위 게임 패키지의 형태의 프로그램을 개발하고자 한다. 마지막으로, 이 프로젝트를 통해 파이썬을 사용한 향후 프로그램 개발 및 심화 프로젝트를 위한 활용 경험과 아이디어를 쌓고자 한다.

**2. 연구 방법 및 과정**

**a. 개발 환경 및 시간**

본 프로젝트는 5월 15일부터 6월 18일까지, 약 1달간 진행하였으며, 전체적인 프로그램 개발은 파이게임을 기반으로 하는 파이썬 코딩을 통해 진행하였다.

**b. 개발 과정(타임라인)**

* 5월 15일: 프로젝트에 삽입할 게임 선정. 예시 및 참고자료 수집.
* 5월 16일: 첫번째 게임 ‘공부수기’ 개발 시작. 게임 구동을 위한 코드 연구 및 자료조사, 구현. 게임 내 사용될 스프라이트 기본 설정
* 5월 17~19일: 공부수기 게임 세부 코드 구현. 장애물의 움직임 설정 및 장애물 파괴 코드 작성.
* 5월 20~23일: 공부수기 게임 레벨 세분화 작업. 레벨에 따른 게임 내 기믹의 설정 및 코드 구현. 시험운전 및 버그 수정.
* 5월 24~25일: 공부수기 게임 편의성 증가. 각 레벨 별 시험운전 및 버그 수정.
* 5월 26~28일: 게임의 구동을 위한 전체적인 프로그램 제작 시작. 화면전환, 게임 전환 방법 구현. 게임 로비, 게임 버튼, 게임 성공 및 실패 화면 구현.
* 5월 29일: 공부수기 게임을 본 게임 프로그램에 이식, 구동 확인 및 버그 수정  
   두번째 게임 ‘슈팅 게임’ 개발 시작.
* 5월 30~31일: 게임 내 사용될 클래스 코드 및 게임 내 기믹 개발. 게임 화면 구현. 운석 클래스 및 게임 충돌 처리 구현
* 6월 1일: 슈팅 게임 코드 충돌 처리 버그 수정. 세번째 게임 ‘테트리스’ 개발 시작.
* 6월 2~5일: 테트리스 게임 내 블록 클래스 구현. 블록 이동 및 회전 코드 구현.
* 6월 6~9일: 테트리스 게임 레벨 세분화 작업. 레벨에 따른 게임 기믹 설정 및 코드 구현. 시험운전 및 버그 수정
* 6월 10~11일: 테트리스 게임 코드 구현 완료. 코드 구동 시험 및 버그 수정. 편의성 증가.
* 6월 12일: 테트리스 게임을 본 게임 프로그램에 이식. 구동 확인 및 첫번째 게임과의 충돌 여부 확인. 버그 수정.
* 6월 13일: 슈팅 게임 충돌처리 버그 수정 및 코드 간소화. 레벨 세분화 작업 및 레벨 별 기믹 구현.
* 6월 14일: 슈팅 게임 시험운전 및 편의성 증가. 본 게임 내 이식 후 구동 및 코드 간 충돌여부 확인.
* 6월 15일: 게임 프로그램 전체 코드 시험운행 및 버그 수정. 편의성 증가.
* 6월 16~18일: 게임의 가시성을 위한 스프라이트 수정 작업. 수정한 스프라이트 본 게임 프로그램에 삽입. 프로젝트 완성

**3. 본론**

1. **게임 틀 프로그램**

프로젝트의 전체적인 형태인, 패키지형 게임 프로그램을 위해 시작 화면, 게임 로비 화면, 각 게임으로 이동하는 버튼을 코드로 구현해 배치하였다. 또한, 각 게임이 실행되기 전과 레벨 전환, 게임 재시도 시의 편의성을 위해 게임 안내 화면과 게임 오버 화면, 게임 클리어 화면을 구현하였다.

1. **버튼 객체**

버튼의 경우, 클래스를 활용하여 일정한 기능을 수행할 수 있는 복수의 버튼 개체를 생성할 수 있도록 설정하였다. 마우스 커서를 버튼에 갖다 놓았을 시 버튼의 텍스트가 변화해 가시성을 높였으며, 버튼 개체 선언 시 게임 내 텍스트, 색상 변화, 이미지 등의 변수를 대입하여 적용할 수 있는 형태로 제작하였다.

**b. 프로그램 시작 화면 및 로비 화면**

프로그램 시작 화면 및 게임 로비 화면의 경우 함수 선언과 함수의 실행 조건 개념을 활용하여 구현하여, 화면 별 표시되는 개체가 달라지도록 설정하였다. 게임 로비 화면의 경우, 버튼 개체를 배치하여 각 버튼을 클릭 시 해당하는 게임 함수가 실행될 수 있도록 설정하였다. 또한, 게임 로비 화면에서 프로그램 시작 화면으로 돌아갈 수 있는 버튼과, 프로그램 시작 화면에서 바로 게임을 종료할 수 있는 버튼을 삽입하여, 창의 닫기 버튼을 누르지 않고 프로그램이 종료될 수 있는 경로를 만들어 프로그램을 자연스럽게 종료할 수 있는 기능을 추가하였다. 이를 통해 게임 내 자연스러운 화면 전환과 편의성을 높였으며, 게임의 시작 화면에서 게임을 종료하는 일반적인 게임 내 UI와 유사한 환경을 구축하였다.

**c. 게임 안내 화면**

게임 버튼 클릭 시 곧바로 게임이 시작되는 점과, 사용자가 게임 내에서 사용되는 키를 알 수 없다는 점, 게임 내 기믹에 익숙하지 않다는 점을 고려하여, 게임 함수가 시작하기 전 게임 방법을 안내하는 화면을 삽입하여 편의성을 높이기로 하였다. 특히 공부수기 게임의 경우, 게임 함수 실행 시 일시정지 및 시간 경과 없이 장애물이 움직이기 때문에 사용자가 게임에 적응할 시간이 부족하다고 판단하였다. 따라서, 게임 로비 화면에서 해당 게임의 버튼을 클릭 시 게임 안내 화면을 먼저 출력함으로써 사용자가 게임의 규칙 및 플레이 방법을 충분히 숙지하고 게임을 시작할 수 있도록 편의성을 높이기로 하였다. 더불어, 게임 안내 화면 내 게임 로비화면으로 돌아갈 수 있는 버튼을 삽입하여 사용자가 각 게임 별 난이도, 게임 방법을 보고 선택할 수 있도록 하였다.

**d. 게임 오버 및 게임 클리어 화면**

각 게임 내 특정 조건을 만족하여 게임이 종료될 경우, 게임 함수가 정지하고 게임의 결과에 따라 게임 오버 혹은 게임 클리어 화면이 출력되는 함수를 구현하였다. 초기 단계에서는 게임의 결과를 변수로 설정하여 입력 받아 출력하는 단일 함수를 구현하려 했으나, 게임 결과에 따라 표시되어야 하는 버튼의 종료와 기능하는 함수가 달라야 하므로 별도의 복수의 함수를 구현하여 실행시키는 것으로 변경하였다. 게임 오버 시에는 해당 게임의 레벨을 다시 도전할 수 있는 기능을 구현하였으며, 해당 레벨을 완료했을 시 다음 레벨의 게임에 도전할 수 있는 기능을 구현하였다. 또한, 게임 로비 화면으로 돌아갈 수 있는 기능을 두 함수에 추가하여 사용자가 언제든지 게임을 종료할 수 있도록 편의성을 높였다.

위의 요소들을 통해 프로그램 내 입력된 다수의 게임 함수 코드를 자유롭게 가동 및 중단시킬 수 있는 환경을 구현하고, 그 과정에서 함수 간 자연스러운 변환이 이뤄질 수 있는 기능을 구현하였다.

1. **공부수기 게임**

첫번째 게임 프로그램으로는, 파이게임의 기본적인 기능 및 이벤트와 파이썬 문법의 조건문, 반복문을 활용할 수 있는 게임을 구현하고자 하였다. 따라서, 캐릭터, 장애물, 무기 간 충돌 조건 설정을 확인하는 기능을 구현해, 게임 내 캐릭터가 발사한 무기로 장애물을 파괴하는 형식의 공부수기 게임을 선택하였다.

**a. 장애물**

무기가 장애물에 충돌했을 경우 장애물의 크기가 작아지거나 두 개의 더 작은 장애물로 쪼개지는 기믹을 채택하였다. 무기 충돌 시 장애물 크기 변경 기믹은 파이썬의 리스트 개념을 활용하여 구현하였으며, 해당 스프라이트가 리스트 내 지정된 이미지로 변경되는 코드를 구현하였다. 또한, 무기와 장애물 객체가 충돌했을 시 무기를 지우는 코드를 구현하여, 게임 내 스프라이트가 많아져 게임이 느려지거나 오류가 나는 것을 방지하였다. 무기와 장애물 충돌로 인해 장애물이 분리된 경우, 장애물이 위로 올라간 후 포물선을 그리며 낙하하는 움직임을 구현해 사용자가 장애물을 피할 수 있도록 하여 편의성을 높였다.

**b. 캐릭터 및 무기**

캐릭터 스프라이트가 게임 화면 내 바닥의 정 중앙에 놓이고, 게임 화면 내에서만 움직일 수 있도록 구현하였으며, 조건문을 사용하여 캐릭터가 장애물에 닿거나, 게임 내 시간 제한이 다 되었을 시 게임 종료 및 더 이상 움직이지 않도록 설정하였다. 무기의 경우, 캐릭터의 정 중앙 부분에서 발사되는 코드를 작성하였으며, 화면의 가장 위에 닿거나 장애물에 닿았을 경우 화면에서 지워지는 조건을 설정하였다. 또한, 가장 작은 장애물의 크기를 고려하여, 위아래로 긴 형태로 설정하여 작은 장애물을 부술 수 있는 형태로 설정하여 편의성을 높였다.

**c. 레벨 세분화**

공부수기 게임의 각 레벨은 장애물이 무기에 충돌했을 시 크기가 작아지는 단계의 변화와 장애물이 분리되는 횟수에 변화를 주어 차별성을 두었다. 첫번째 레벨의 경우 무기로 격추 시 장애물의 크기만 작아지는 기본적인 기믹을 설정하였다. 두번째 레벨에서는 첫번째 장애물 격추 시 장애물 스프라이트가 2개의 작은 장애물로 분리되는 기믹을 설정해 난이도를 높였다. 또한, 화면 양쪽에 벽을 두어 장애물 스프라이트가 움직이는 공간을 좁혀 난이도를 높임과 동시에, 캐릭터가 공을 피할 수 있는 공간을 만들어 난이도를 조정하였다. 마지막 레벨은 장애물 스프라이트가 분리되는 단계를 늘려 최대 8개의 장애물이 화면에 나타나도록 난이도를 높였다.

이러한 과정을 통해 파이썬의 기본 문법인 조건문과 반복문을 활용하여, 게임 내 기믹을 다양화시키고, 변수의 활용을 통해 편의성과 난이도를 조절하여 프로그램의 세부사항을 조정 및 수정하였다.

1. **슈팅게임**

두번째 게임으로는 파이썬 문법 중 클래스의 개념을 활용하여, 다수의 스프라이트가 화면에 등장하고, 각 스프라이트의 상태를 지속적으로 처리할 수 있는 프로그램을 구현하고자 하였다. 더불어, 공부수기 게임에서 사용하였던 코드의 구조를 참고 및 활용하여 다른 형태의 게임을 만들고, 클래스의 개념을 적극 활용해 코드를 간소화시키고자 하였다.

1. **전투기 및 미사일 객체**

공부수기 게임에서 사용하였던 캐릭터 객체의 코드를 활용하여 움직임을 구현하였으며, 클래스를 적용하여 코드 파악 및 레벨 별 전투기 객체의 속도 조절이 용이하도록 설정하였다. 미사일의 경우 전투기 스프라이트의 중앙에서 발사되도록 위치를 설정하였으며, 발사 시 해당 키를 눌렀을 시 한발씩 발사되고, 리스트를 설정하여 여러 개의 미사일이 화면 내 출력될 수 있도록 코드를 구현하였다. 또한, 클래스 내 직접 충돌 함수 및 조건을 설정하여 충돌 시 조건 설정 코드를 간소화할 수 있도록 개선하였다. 전투기는 운석과 충돌하였을 경우 게임 오버 화면이 출력되며 게임이 중단되는 조건을 설정하였으며, 미사일은 운석 객체와 충돌 시 화면에서 지워져 전체 프로그램이 느려지거나 오류가 생기는 것을 방지하였다.

1. **운석**

운석 객체는 클래스를 활용하여, 화면 내 다수의 독립된 운석 객체가 출력될 수 있게 구현하였으며, 이를 통해 화면 내 사용자가 고려해야 할 스프라이트의 수를 늘림으로써, 위의 공부수기 게임보다 난이도를 높이면서 전체적인 게임 기믹의 차별성을 두었다. 또한, 클래스 내 속도 변수를 지정하고 운석의 속도 범위를 설정하여, 동일한 화면 내의 각 운석 객체의 낙하 속도가 다르게 적용될 수 있도록 코드를 작성하여 게임 기믹이 단순화되는 것을 방지하였다. 운석의 모양은 리스트와 랜덤 모듈을 사용하여 리스트 내 무작위 이미지가 운석에 적용되어 출력될 수 있도록 설정하였으며, 이미지의 넓이에 따라 미사일과 충돌 처리되는 구역이 달라지도록, 각 이미지의 넓이에 get\_rect를 적용시켜 코드를 구현하였다. 또한, 운석의 발생 확률 및 이에 따른 운석 생성 조건문을 설정하여 운석이 화면에 출현하는 빈도수를 조절하였다.

1. **레벨 구성**

각 게임에서 운석을 파괴하지 못하고 놓친 횟수가 3번이 되었을 경우 게임 오버 화면이 출력되는 조건을 설정하였다. 또한, 사용자가 놓친 횟수를 쉽게 확인할 수 있도록, 놓친 횟수를 그대로 표기하는 것이 아닌, 놓친 횟수 변수와 목숨 변수를 설정하여 이를 토대로 게임 내 실패 횟수만큼 목숨이 차감되는 코드를 구현하여 화면 우 상단에 출력하도록 표시하였다. 더불어, 게임의 승리조건을 일정 점수 달성이 아닌, 일정 시간 생존으로 설정하여, 사용자가 빠른 시간 내 레벨을 완료하고, 각 레벨을 자연스럽게 완료 및 종료할 수 있도록 설정하였다.

각 레벨에서는 운석의 낙하 속도 및 생성 빈도수를 증가시키는 방식으로 난이도를 점차 증가시키도록 하였다. 두번째 레벨에서는 운석의 생성 확률과 레벨 완료를 위한 시간을 증가해 난이도를 조정하였다. 세번째 레벨에서는 운석의 낙하 속도, 생성 빈도수, 레벨 완료를 위한 시간을 증가시켰다. 또한, 세번째 레벨의 특수 기믹으로, 미사일로 파괴 시 즉시 게임 패배 조건이 달성되는 탈출포드 객체를 추가하여 기존 레벨과의 차별점을 두었다. 탈출포드 객체는 운석 객체보다 빠른 속도로 떨어지며, 전투기와의 충돌 판정이 발생하지 않게끔 설정하여, 사용자가 운석을 파괴하는 데 용이하도록 난이도를 조정하였다. 또한, 전투기의 이동 속도를 증가시켜 운석의 낙하 속도 및 생성 확률 증가로 인한 과도한 난이도 상승을 방지하였다.

위 과정을 통해 파이썬의 클래스 객체를 활용하여 다수의 독립된 객체를 일괄적으로 관리 및 처리하는 기능을 제작하였으며, 코드를 간소화하여 오류 및 개선 사항을 찾아내는 데 용이한 구조를 구현하였다. 또한, 각 레벨 별 운석의 출현 빈도수 및 속도의 변화, 게임 내 고려할 조건의 추가를 통해 각 레벨을 차별화시키고 게임 기믹에 다양성을 주었으며, 게임 실행 시 사용자가 난이도 상승으로 인한 피로감을 느끼지 않도록 각 객체의 세부 속성을 조정하였다.

1. **테트리스 게임**

세번째 게임으로는 함수와 클래스 객체를 활용하되, 기존의 두 게임과는 구조의 코드를 사용하여 더욱 복합적인 기능을 구현하는 게임 코드를 제작하고자 테트리스 형 게임을 선정하였다.

1. **블록 객체**

블록 객체는 클래스를 활용하여 다양한 모양의 블록이 나올 수 있는 기능과, 복수의 스프라이트가 화면에 표시 및 처리될 수 있는 기능을 구현하였다. 또한, 블록이 출력되는 범위를 한정하고, 각 줄은 리스트를 이용하여 블록이 쌓인 상태와 빈 상태를 확인하는 코드를 작성해, 블록이 바닥에 차례대로 쌓이는 구조와, 한 줄이 모두 채워졌을 경우 그 줄이 지워지면서 점수가 올라가는 코드를 구현하였다.

1. **레벨 구성**

각 레벨은 블록이 표시되는 화면의 가장 윗부분까지 쌓였을 경우 게임이 종료되며, 일정 점수를 획득했을 시 승리 조건이 달성되도록 설정하였다. 게임의 레벨이 높아질수록, 승리 조건 달성을 위한 점수를 증가시키는 방법으로 난이도를 조절하였다. 첫번째와 두번째 레벨에서는 낙하하는 블록 다음에 나올 블록의 모습을 사전에 보여주는 기능을 구현하여 사용자가 다음 블록을 배치할 자리를 미리 고려할 수 있도록 하였다. 세번째 레벨의 경우, 레벨 클리어를 위한 점수를 증가시켰음에도 체감상 난이도에 크게 차이가 없는 모습을 보였기에, 이를 해결하고자 다음 블록이 배치되는 부분을 이미지로 가려, 사용자가 다음 블록을 배치할 위치를 미리 계산할 수 없도록 하여 난이도를 조정하고 기존 레벨과의 차별성을 주었다.

위와 같은 과정을 통해, 클래스를 통해 생성한 개체를 화면 내에서 지속적으로 확인하여 처리하는 과정을 구현하였으며, 게임 내 기믹의 다양화를 통해 게임이 단순한 동작의 반복적 실행이 되지 않도록 하였다.

**4. 결과 및 결론**

1. **프로젝트 실행 결과**

공부수기 게임, 슈팅게임, 테트리스 게임의 총 3가지의 게임을 구현하는 것을 통해, 파이썬 세미나 및 관련 서적을 통해 익힌 파이썬 코드의 구조의 활용에 대해 직접적으로 연구하고 이를 구현하는 데 성공하였다. 각 게임의 레벨 별 조건을 설정하여 이에 대응하는 함수가 실행되는 기능을 구현하였으며, 레벨 별 다른 변수가 적용되어 난이도 및 기믹의 변화를 주는 데 성공하였다. 또한, 세 게임을 포함 및 실행하는 전체 게임 프로그램을 작성하여, 서로 다른 게임의 변수 및 함수가 충돌하지 않고 독립적으로 실행될 수 있는 기능을 구현하였으며, 각 게임의 자연스러운 실행 및 전환을 통해 사용자의 편의성을 최대한 증가시켰다. 그러나 코드 구축 과정에서 발생한 변수 간 충돌 및 오류로 인하여 초기에 계획한 프로그램의 형태를 완벽하게 구현하지는 못하였다.

**2. 프로젝트 개선점**

1. 테트리스 게임을 전체 게임 프로그램에 이식하는 과정에서, 레벨을 재시작 하거나 다음 레벨로 넘어갔을 시, 처음 블록이 자동으로 내려오지 않아 직접 키를 조작하여 내려줘야 다음 블록이 자동으로 내려오게 되는 오류가 있었으나, 몇 차례의 시도에도 해결하지 못하였다. 추후 프로젝트에서 함수의 구조를 개선하여 해당 오류를 해결해야 할 필요가 있다. 또한, 테트리스 블록의 낙하 속도를 조절하여 레벨의 난이도를 조절하려 했으나, 이 과정에서 낙하속도 변수가 서로 충돌하는 문제가 발생하여 이를 구현하지 못하였다. 추후 프로젝트에서는 이를 개선하여 테트리스 게임의 각 레벨 간 난이도의 유의미한 차이가 있도록 구현할 필요가 있다.
2. 슈팅 게임의 경우, 다수의 운석을 처리해야 하는 게임의 특성을 고려하여 게임 화면의 가로축을 좁게 설정해야 하는 사항이 존재하였다. 이로 인해 전체 화면의 넓이를 다른 게임의 화면과 동일하게 설정하되, 슈팅 게임 실행 중에는 게임 실행 화면을 전체 화면의 중앙 부분에 한정하려 했으나, 화면의 경계 부분에서 낙하하는 운석 스프라이트가 깨지면서 낙하하는 오류가 발생하였다. 그리하여, 슈팅게임 실행 중에는 화면의 넓이를 게임 함수 내에서 별도로 설정하여 전체 화면의 넓이가 줄어든 상태로 게임이 안정적으로 실행되게끔 설정하였다. 또한, 운석 클래스 객체의 이미지 파일을 불러오는 과정에서 예기치 못한 오류가 발생해, 부득이하게 이미지 경로를 절대경로로 설정해 게임 실행의 편의성을 저하시키게 되었다. 추후 프로젝트에서 함수의 구조를 개선하여 해당 사항을 개선할 필요가 있다.
3. 전체적인 게임의 구조에 있어, 게임의 레벨 세분화 및 레벨 전환, 각 레벨 별 독립적인 기믹을 구현하는 과정에서, 별개의 다수의 함수를 제작해 구현하게 되면서 코드의 구조가 다소 복잡하고 길어지게 되었다. 향후 프로젝트에서는 적절한 변수 및 클래스를 활용하여 최소한의 함수를 통해 레벨 세분화 기능을 구현할 필요가 있다.
4. **3. 향후 연구방향**
5. 향후 연구에서는 해당 프로젝트에서 제작했던 게임 프로그램 내 기믹을 다양화하는 방향으로 나아가고자 한다. 시중의 모바일 게임 내에서 보이는, 돌발적인 장애물의 등장, 두 번 격추해야 파괴되는 장애물을 등장시키는 등, 기존의 게임의 요소를 다양화하여 역동적인 프로그램을 개발하고자 한다. 또한, 게임 내 대부분의 스프라이트가 이미지의 경계선이 아닌, 이미지 파일의 넓이로 충돌처리가 되는 상태이기에, 향후 프로젝트에서는 이를 해결할 수 있는 방법을 강구하여, 게임의 가시성을 높이고 사용자가 게임을 실행하는 데 있어 편의성을 증가시킬 필요가 있다. 마지막으로, 해당 프로젝트에서 제작한 게임과 다른 구조로 기동하는 다양한 종류의 게임을 제작해 프로그램 내 더하고자 하며, 게임 결과에 따른 랭크 측정 시스템, 일정 점수 달성 시 다음 게임이 해제되는 기믹을 구현하고자 한다.

**4. 프로젝트 의의**

1. 이 프로젝트를 통해 파이썬 코드의 활용을 통해, 플래시 게임과 유사한 형태의 게임 프로그램을 제작하고, 변수, 함수, 클래스의 활용에 대한 이해도를 높일 수 있었다. 또한, 플래시 게임과 같은 간단한 구조의 게임에서 객체의 움직임 및 조건 설정에 대하여 이해할 수 있게 되었으며, 향후 이를 활용해 더욱 복잡한 형태의 프로그램을 제작하는 데 있어서 필요한 지식과 통찰력을 얻을 수 있었다. 더불어, 파이썬 코딩을 통해 터미널 내에서만 결과를 확인하는 프로그램에서 한단계 나아가, 코드의 결과를 가시화할 수 있는 프로그램을 제작할 수 있었다. 또한, 본 프로젝트의 코드 활용을 통해 추후 프로젝트 기획 및 진행에서 어플리케이션 제작, 아두이노를 이용한 기기 제작에 참고할 수 있는 프로그램을 제작해보고, 사용자의 편의성을 고려한 코드를 구축할 수 있었다.

**5. 부록**

**참고문헌**

다나카 겐이치로. *게임으로 배우는 파이썬.* 영진닷컴. 2019. p.257 ~ 264.

나도코딩. “파이썬 코딩 무료 강의 (활용편1) - 추억의 오락실 게임을 만들어 보아요. 3시간이면 충분합니다. [나도코딩]”. 나도코딩. 2020. 5. 3. 동영상. 2:52:34. https://youtu.be/Dkx8Pl6QKW0.

이수안. “파이썬 비행기 슈팅 게임 만들기 ver.2 Python Shooting Game ver.2”. 이수안컴퓨터연구소. 2020. 6. 28. 동영상. 58:58. https://youtu.be/W92RjjptAsM.