퍼즐 카우(Puzzle Cow)

프로젝트 명	퍼즐카우(Puzzle Cow) – Curse를 활용한 슈팅 퍼즐게임	
프로젝트 팀원	김경수	2017114038
	류진용	2017110849
	윤 진	2017111908
	정명원	2017111995

과목명	오픈소스 프로그래밍
담당교수님	류은경 교수님
제출날짜	2018.06.20

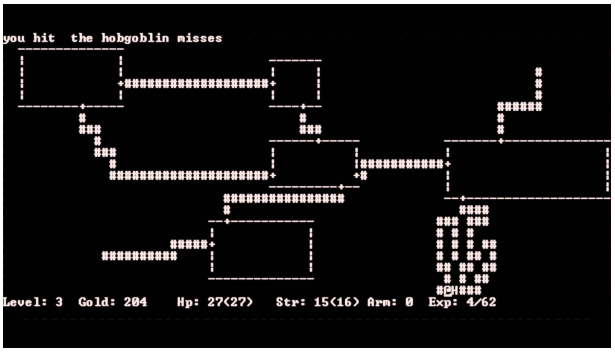
목차

1.1 프로젝트 주제

수업시간에 배운 curse기능을 활용하여 다양한 레벨과 기능이 있는 슈팅 퍼즐게임을 GitHub를 통해 협력하여 제작

1.2 주제 선정 이유

오픈소스 7강의 주제인 curse기능은 기본 프로그램만 실행하는, 단조롭게만 보였던 터미널에서 역동적인 화면을 출력할 수 있다는 점이 흥미로웠습니다. 그래서 이번에 오픈 소스 프로젝트가 주어졌을 때 제일 처음 떠오른 것이 바로 curse 기능이었습니다. 이 curse 기능을 통해 어떤 것을 만들 수 있을까 생각하던 도중에 책에서 curse 기능을 이용해 만들어낸 pong 이라는 게임을 발견했습니다. 이에 대한 설명을 보고 이번 프로젝트에서 터미널을 이용해 게임을 만들어보는 것이 어떨까 하는 아이디어가 등장했고 회의 끝에 그 아이디어가 채택되면서 게임을 만드는 것으로 결정되었습니다. 그 후 터미널에서 할 수 있는 여러 가지 게임을 생각해보았는데, 처음에는 Rogue나 스네이크 같은 텍스트 위주의 간단한 게임을 구상했습니다. 그것도 마음에 들었지만 새로운 무언가를 만들어 보고 싶었습니다. 또한 터미널에 색깔을 넣는 기능을 구현하게 되면서 새로운 것을 만들 발판을 마련했습니다. 결과적으로 어릴 적 학교 앞 오락실에서 즐겨했던 퍼즐버블에서 아이디어를 따와 터미널에서 하는 퍼즐버블을 만들어보자는 목표로 이번 프로젝트를 구성하게 되었습니다.



-Rogue

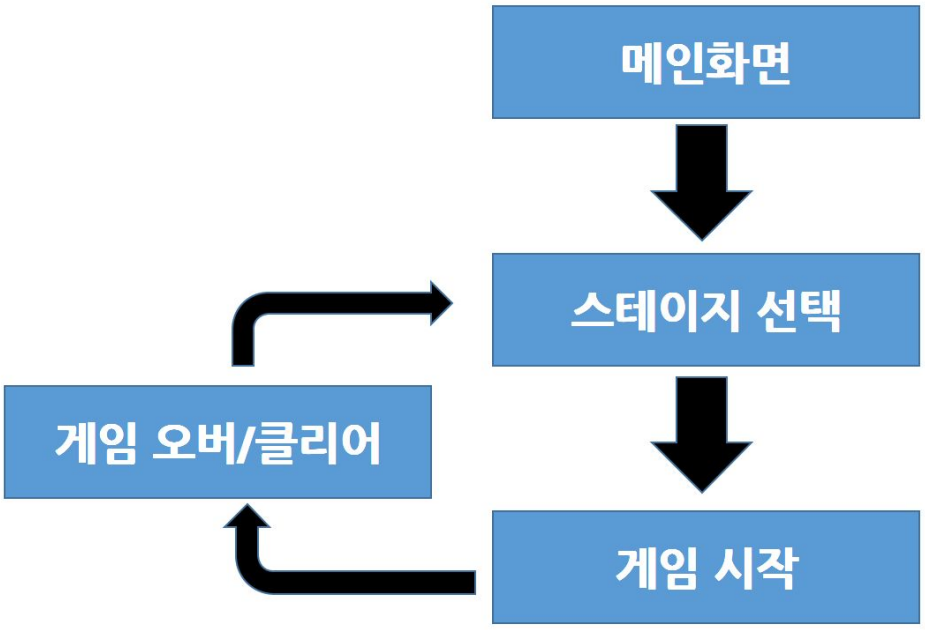
-Puzzle Bobble

1.3 프로젝트 목표

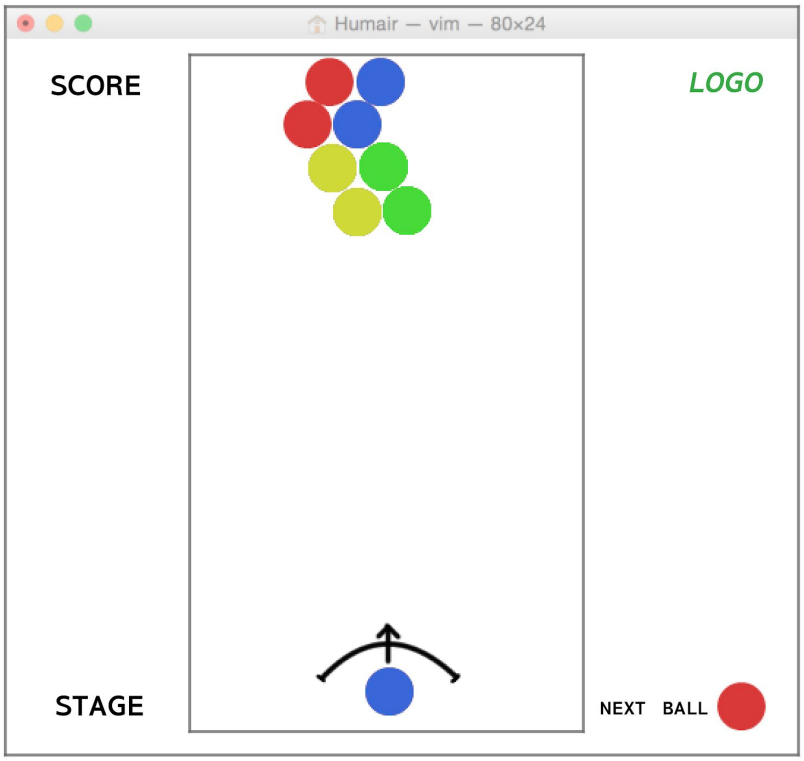
오픈소스 프로젝트라는 이름에 걸맞게 오픈 소스를 어떻게 사용할 것인지 어떻게 소스를 공유하는지를 알아보는 것이 목표입니다. 남들에게 소스를 공유하기 위해서는 자신이 짠 코드가 얼마나 남들에게 잘 읽히는지가 중요할 것입니다. 단순히 주석을 붙이는 것 뿐 아니라 변수 명을 알기 쉽게 바꾼다던지 함수는 그 함수의 이름에 맞는 역할만을 수행하도록 하는 등의 과정을 겪으면서 남들에게 잘 읽히는 코드를 작성하는 능력을 기를 수 있을 것이라고 생각합니다. 또한 Git 과 Github 그리고 거기서 파생된 Git Kraken을 써보면서 이런 오픈 소스를 팀원들 간에 어떻게 공유하고 관리해 나가는지 배우는 것이 목표입니다.

터미널에서 가능한 여러 기능들을 사용해서 퍼즐 버블의 특징들을 잘 품은 완성도 높은 게임을 만들어 내는 것입니다. 여러 가지 스테이지를 만들어 다양한 플레이의 기회를 제공합니다. 또한 퍼즐버블에서 지원하는 점수 측정을 추가해 플레이어에게 성취감을 주도록 합니다. 배경음악 재생 기능을 추가하여 게임에 몰입할 수 있도록 하는 것이 목표입니다. 이런 다양한 기능들을 추가하여 더 다양한 기능을 제공하는 게임을 만듭니다.

1.4 프로젝트 계획 범위



〈 프로젝트 계획 단계의 프로그램 목표 구상도 〉



1.4.1 맵 외부 환경들

실제 공이 튀는 맵 바깥을 살펴보면 왼쪽 상단에는 현재까지의 스코어를 표시해주는 기능을 넣어 플레이어에게 동기를 부여하는 역할을 줄 것입니다. 그리고 왼쪽 하단에는 현재 스테이지와 최고 스테이지를 표시해줍니다. 또한 오른쪽 상단에는 로고를 표시하였고 오른쪽 하단에는 다음에 나올 공을 미리 알려주어 현재 어떻게 플레이하는 것이 다음 상황에서 좋을지 알도록 하는 것입니다.

1.4.2 맵 내부 환경

실제 공이 튀는 맵 안에는 각도를 계산하여 이동경로를 표현해주고 같은 색 공과 마주쳤을때 혹시 3개 이상이라면 그 공들의 파괴와 그 파괴된 공 아래에 있는 공들 을 모두 파괴하는 작업을 진행하고 혹시 3개보다 적거나 다른 색깔의 공이라면 그 공 옆에 붙어 다음 공이 오기를 기다리도록 합니다.

1.4.3 게임 전체적 환경

플레이 화면 이외에도 메인 메뉴를 만들어 지금까지 클리어한 스테이지들을 표시해주고 클리어한 경우 스테이지를 선택할 수 있게 만들어 똑같은 플레이를 반복해야하는 수고를 덜었습니다. 또한 메인 메뉴에 커다란 로고를 만들어 저희가 하고 있는 활동을 확실하게

표시하였습니다. 그리고 배경음악을 추가해 사용자에게 긍정적인 효과를 줄 수 있도록 했습니다.

1.5 프로젝트 환경

- OS : Mac Terminal / Tool : Visual Studio 혹은 Xcode 등 / GitHub(Git Kraken)

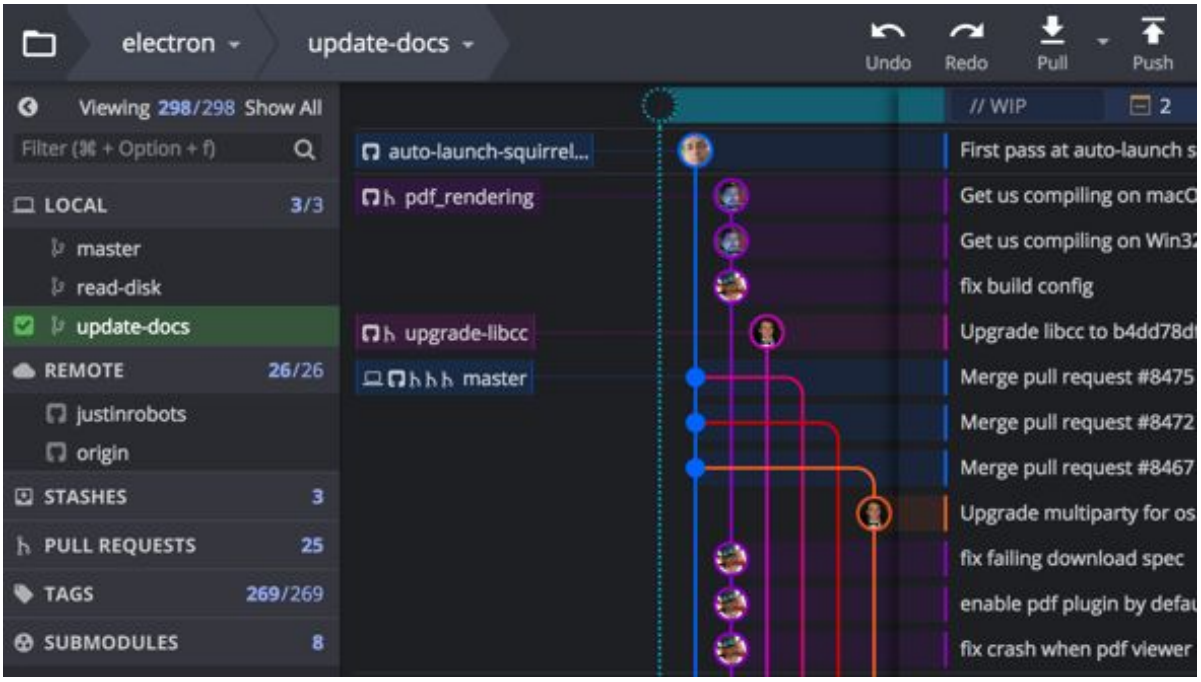
1.5.1 OS 와 Tool

저희가 지금까지 수업시간에 다뤘던 코드들이 MAC에서 작성되었고 terminal에서 사용해야할 코드를 만들어야 해서 Mac OS를 사용하였고 Tool은 각자에게 편한 coding용 tool을 사용할 수 있도록 하였습니다. 혹시 Visual Studio 2017 버전에서 사용되는 _s 메소드들을 사용해야할 경우 전부 기존의 메소드로 교체할 수 있도록 _CRT_SECURE_NO_WARNINGS 를 define하여 기존의 메소드들을 사용할 수 있도록 하였습니다.

다양한 오픈 소스들이 Windows로 제공되고 있다는 점이 큰 단점으로 다가왔지만 MAC에서 적응한 만큼 MAC용 코드를 찾아내는 과정을 거쳐 좋은 소스들을 찾아냈습니다. 또한 다양한 MAC 에서 사용되는 함수들을 찾아내면서 새로운 기능들을 구현하는데 도움을 얻었습니다.

1.5.2 GitHub

협력 작업을 위해서 GitHub를 사용했는데 터미널에서 GitHub사용은 너무 보기 힘들고 실제로 어떻게 작업이 진행되고 있는지 알 수 없었습니다. 예를 들어 현재 브랜치가 어떤 것이 있으며 어떻게 나뉘어있는지, 누가 어떤 내용을 push했는지 보기 힘들었습니다 또한 어떻게 commit이 이루어 졌는지 merge는 어떤 방식으로 진행되었는지를 보기 힘들었는데 이러한 점을 Git Kraken이라는 프로그램을 추가로 사용하여 진행하였습니다. Git Kraken에서는 다양한 merge branch push 등의 작업을 점과 그 점들을 잇는 선으로 좋은 GUI를 가지고 있는 소스코드 관리 프로그램이였습니다. 하지만 GitHub 자체적인 문제로 push나 pull이 제대로 되지 않거나 적용이 늦게 되어 코드가 꼬이는 경우가 종종 발생하였습니다.



- Gir Kraken 예시 사진

2. 프로젝트 수행

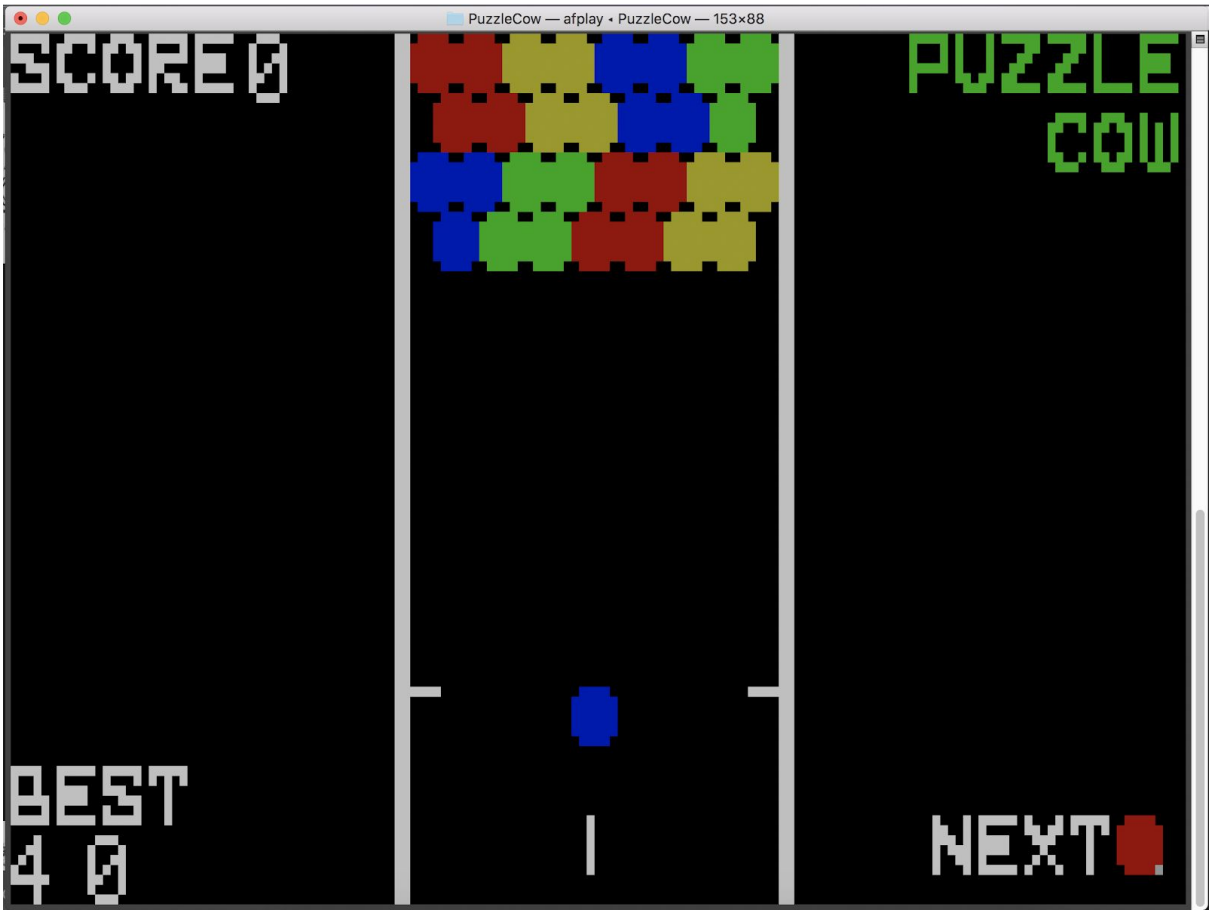
2.1 프로젝트 수행 내용

2.1.1 메인화면

- 실제 프로젝트 속 메인화면

메인화면의 정중앙에는 저희 프로젝트의 이름인 Puzzle Cow를 로고로 만들어 새겼고 그 아래에 현재 진행할 수 있는 스테이지 목록들이 나열되어 자신이 원하는 스테이지를 플레이할 수 있도록 했습니다. 이 숫자들은 클리어한 양에 따라 정해지는 것으로 예를 들어 2스테이지까지 클리어했을 경우 메인화면의 숫자는 1,2,3 까지만 등장하고 4는 등장하지 않습니다. 그래서 제일 처음 시작한 플레이어의 경우에는 1의 숫자만 등장해 첫 스테이지부터 게임을 진행하게 됩니다. 여기서 부터 배경음악이 나오기 시작합니다.

2.1.2 게임 화면



- 실제 프로젝트 속 게임 화면

게임이 시작될 경우 현재 선택한 스테이지에서 공이 올라가고 게임 화면으로 넘어가게 됩니다. 이 게임 화면에서 왼쪽 상단에 있는 것은 현재 스코어를 표시하고 왼쪽 하단에는 최고 어느 스테이지까지 도달했는지를 나타냅니다. 그리고 오른쪽 상단에는 로고가 표시되어있고 오른쪽 하단에는 다음 나타날 공의 색깔을 보여줍니다. 그리고 중앙에 있는 화면에서 위에 있는 공 뭉치들은 스테이지에서 파괴해야할 공뭉치들이고 전부 파괴할 경우 다음 스테이지로 넘어가게 됩니다. 스테이지 별로 공뭉치의 배치가 달라 클리어하는 방식이 다를 것입니다. 그리고 아래쪽의 파란 공은 현재 발사할 공의 색깔을 나타내고 그 파란 공 왼쪽에 있는 선은 이곳까지 공이 닿을 경우 게임에서 지게 되는 한계선의 위치를 나타냅니다. 그리고 공 아래에 있는 선은 현재 공이 이동할 방향을 나타내는 것으로 좌우 방향키를 사용하여 각도를 조절 할 수 있습니다.

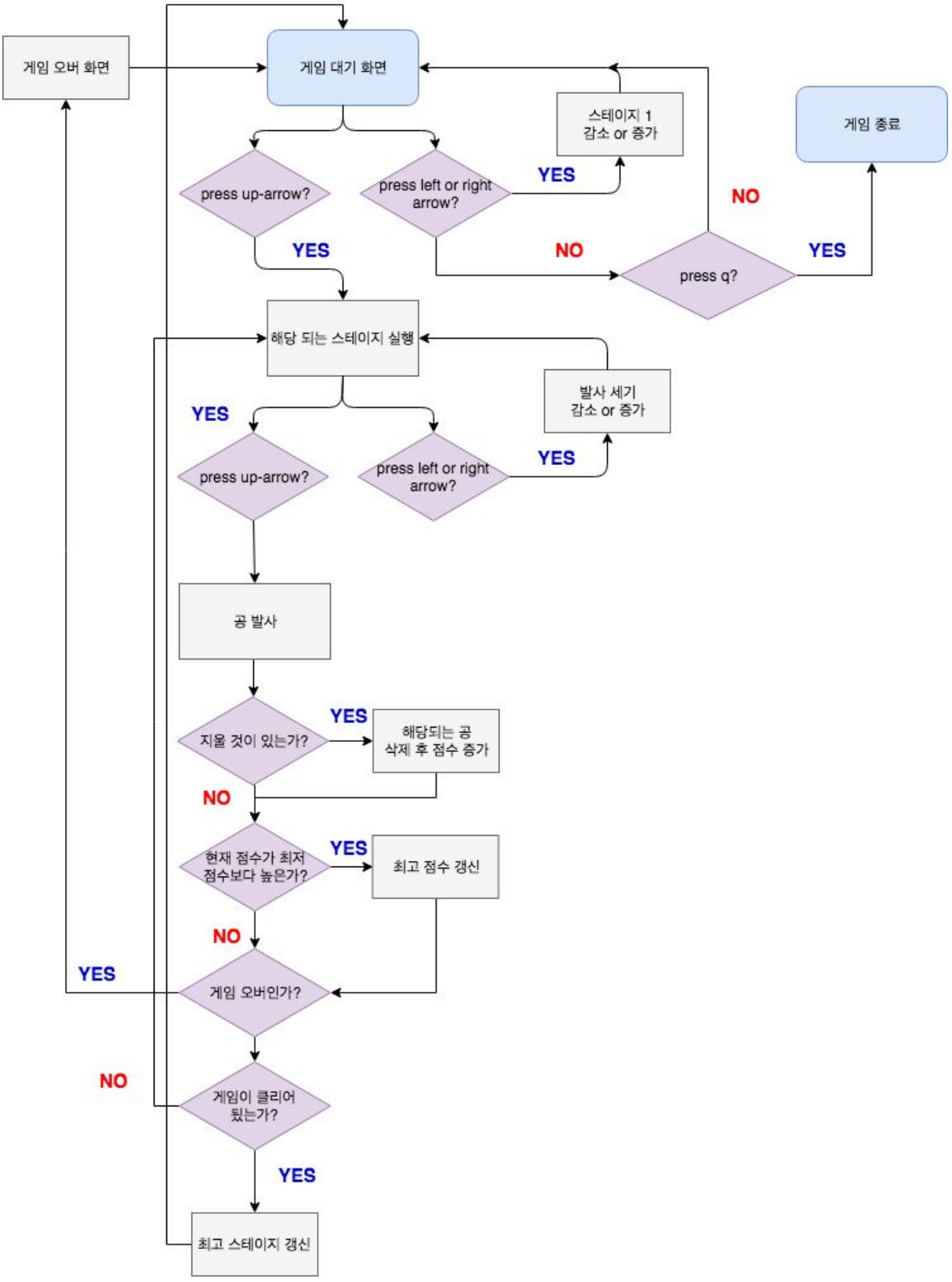
게임이 끝날 경우 전부 파괴를 해서 게임이 끝난다면 다음 스테이지로 넘어가게 되지만 한계선에 도달하여 게임이 끝난다면 초록색 개구리 형상을 보여주어 슬픔을 표현하도록 했습니다.

2.2 개발 과정 및 시행착오

저희 조는 이번 프로젝트를 개발하면서 Lean적인 방법을 적용해 보았습니다. Lean 적인 방법이란 Build -> Measure -> Learn의 Cycle을 돌면서 지속적인 변화를 주는 방식입니다. 저희 조에서 처음으로 기획하여서 프로젝트를 시작한 아이디어는 “리듬 게임”이었습니다. 노트 처리와 관련된 시간별로 값을 처리하는 것이나 여러개의 쓰레드로 음악을 재생시키는 등등의 기능을 구현하려고 기획하였습니다. 그 후, 역할을 분담하여 리듬 게임 프로젝트를 시작하였습니다. 하지만 여러 명이서 각자 다른 코드를 코딩한 후 merge 하는 과정에서 변수나 함수 명 등등의 이유로 다양한 오류가 발생되었습니다. 이런 부분에 있어 다인이 함께 개발할 때에 소스 코드 관리의 중요성을 인지하게 되었습니다. 그래서 가급적 GitHub를 사용하려고 하였습니다. 그리고 리듬 게임에서 가장 중요한 음악 재생이 순탄치 않았습니다. 일반적으로 콘솔에도 사용되는 “FMOD” 라이브러리를 사용하고자 하였지만, 일반적으로 FMOD를 사용할 때에는 윈도우 환경에서 작업을 하는 경우가 많았습니다. 그래서 Mac에서는 FMOD 함수가 정확히 작동이 되지 않는 오류가 생겼습니다. 이를 해결하기 위해 Mac용 IDE인 XCode에서 path값을 설정하는 등, 기본 설정을 변경해 보았지만 여전히 문제가 발생하였습니다. 그래서 지속적인 발전이 불가능하다고 판단하여 새로운 아이디어를 정하기로 하였습니다. 다시금 아이디어 회의를 거치면서 새로운 아이디어를 정하였습니다. 그것이 바로 “Puzzle Cow”입니다. 그리고 지난 시행착오를 바탕으로 저희는 새로운 프로젝트를 기획하기 위해 새로운 방식인 Lean을 채택하였습니다.

여러 명이서 작업을 하여 코드가 꼬이는 문제가 발생하는 것을 막기 위해, 소스 코드 관리 프로그램인 Git Kraken을 활용하여서 그 부분을 방지하였습니다. 또한, 빈번한 회의를 통해서 유동적인 개발을 채택하였습니다. 회의에서 현재까지의 자기의 개발 현황과 함수의 내용을 설명해주거나 앞으로 어떤 함수를 디자인 중인지 등등을 공유하여서 최대한 조원들이 하나의 집단이 되고자 노력하였습니다.

아래는 저희가 실제로 진행했던 프로젝트에 적용한 Lean의 과정들입니다.
아이디어 회의 -> 개발 -> 기존 아이디어 폐기 -> 퍼즐 카우로 전향 -> 초기 프로토타입 완성 -> 개선 및 회의 -> 개선 및 회의 -> 개선 및 회의 -> 유지보수



< PuzzleCow의 흐름도 >

3. 프로젝트 수행 소감

<정명원>
팀단위 프로젝트를 할 때 대규모의 소스 코드 관리를 하는 방법을 배울 수 있었습니다. 보통 사람들이 사용하는 윈도우나 유닉스 환경이 아니라 맥 환경에서 구현하느라 사용하지 못하는 함수도 있었지만, 맥에서만 사용할 수 있는 함수를 사용하기도 하였습니다. 수업시간에 배우지 못한 것도 팀원들과 함께 방법을 찾아나가고 구현해 나가는 과정이 오픈소스프로그래밍 수업을 이해하는데 도움이 되었습니다.

<류진용>
무엇보다 4인 이상의 팀 프로젝트를 경험할 수 있다는 점이 크게 작용된 것 같습니다. 많은 양의 코드와 그것을 짜는 많은 사람들을 조화롭게 잘 엮을 수 있는 역량이 필요하다는 것을 느꼈습니다. 이 때의 역량은 비단 팀장만의 역량이 아닌 개개인의 역량도 크게 작용함을 느꼈습니다. 그래서 이것을 방지하고자 잦의 회의와 의견 교류를 통해서 최대한 완화하는 과정을 채택하였는데, 소스 코드 관리에서 뿐만이 아니라 다른 긍정적인 영향도 많이 받았다고 생각합니다. 또한, 소스 코드 관리에서 더 느낀점이 있다면, Git karken에 오류가 다수 있어 제 계정에서는 오류가 잦게 떠서 커밋이나 푸쉬, 풀이 제대로 적용되지 않은 점이 아쉬웠습니다. 때문에 Git적인 지표가 명확히 들어 나지 않았던 것 같습니다. 하지만 그런 것을 감안하고도 소통이나 대형 프로젝트를 경험할 수 있어 좋았습니다.

<윤진>
이번 개발을 하면서 가장 어려웠던 점은 뭘 해야할지 모르겠다는 점 이었습니다. 그래서 팀원간의 소통이 중요했고 여기서 역할 배분이 중요하다는 것을 알 수 있었습니다. 전에 봤던 글에서 성공적인 협업을 위해 필요한 조건들에 대한 항목들을 본 적이 있었는데 그중에 기억나는 항목을 적어 보자면 모두 동등하게 참여하도록 격려하기, 회의에서 들리지 않는 목소리를 키우기, 자기 자신과 타인에게 책임감 갖기 등 이 있었으며 이번 팀 프로젝트에서 잘 지켜졌었던 항목이라고 생각합니다.
깃 활용에 익숙하지 않아 GUI인 Git Karken을 사용하였는데 이마저도 제대로 관리가 어려워 류진용 학우의 코드를 대신 푸시 해 주기도 하였습니다. 프로젝트가 끝나갈때 쯤 사용법을 익힌 것 같아 이번에 깃을 제대로 활용하지 못한것에 큰 아쉬움이 남았습니다.

<김경수>
저는 GitHub와 GitKarken이라는 프로그램을 처음 써보는데 사실 불편한 점이 많았습니다.(물론 창의융합설계라는 과목에서 사용한 적이 있긴 하지만 그 때는 코드를 쓰기만 했고 그것을 다운받아 수정해서 push하는 행위를 한 적이 없습니다.) 이번에 프로젝트를 진행하면서 제대로 push가 되지 않거나 제가 했던 push가 상대방에게 넘어가는 사건이 있기도 했거든요. 하지만 다른 프로젝트를 진행하면서 소스코드를 이런 프로그램에 맡기지 않고 직접 파일을 주고 받는 경우가 있었는데 그 때마다 이렇게 받은 코드들을 어떻게 합쳐야될지 혼란이 오기도 하고 코드를 붙여넣다가 실수를 하는 바람에 버그가 발생해 그 때문에 며칠을 고생하기도 했습니다. 그래서 소스코드를 관리하기 위해서 이러한 소스코드 관리 프로그램의 필요성을 깨닫게 되는 계기가 되기도 했습니다. 아직도 능숙하게 사용한다고는 말할 수 없지만 GitHub를 사용하는 기본적인 방식을 이해하는 계기가 되었습니다. 또한 프로젝트를 진행하면서 각자 역할을 분담하여 각자의 업무를 진행하고 서로 모르는 부분이 있다면 도와가는 과정을 통해 어떻게 하면 협력할 수 있는지에 대해서 알게 되었습니다.그리고 여러 번의 회의를 통해 의견을 도출해내면서 상대를 설득시키는 방법이나

상대의 의견에 동조하는 방법등의 소통능력이 좋아진 것도 같습니다. (물론 아직 부족합니다만..) 또한 프로젝트에 대한 내용을 진행하면서 오픈 소스와 터미널 프로그램에 대한 이해가 높아졌습니다.