Memory Game

과제제목: Memory Game

제출일자: 2021년 1월 18일 (월)

학 교: 광운대학교

학 과: 컴퓨터정보공학부

성 명: 김 호 성

1. Introduction

유니티를 사용하여 C#스타일로 Memory Game을 구현한다.

Memory Game은 무작위의 카드 더미들 중 짝을 찾아내어, 모든 카드를 뒤집는 게임이다.

Scene별 설명

1. Menu

- MenuBehavior Scripts를 사용하여 Scene 전환을 할 수 있도록 한다.

각각의 버튼에는 triggerMenuBehavior 함수가 들어가 있다.

Start 버튼을 누를 경우 0

Best Score 버튼을 누를 경우 1

Quit Button을 누를 경우 2

Game History 버튼을 누를 경우 4라는 값을 전달한다.

- Start => Level1 Scene으로 이동한다.

- Best Score => Best Score Scene으로 이동한다.

- Quit => 프로그램을 종료한다.

- Game History => Game History Scene으로 이동한다.

2. Level1~3

- Card: Level 별 카드의 수는 4, 8, 18장이며, 카드의 뒷면의 색상을 다르게 구현했다. 또, 카드의 앞면의 경우는 52장을 두었으며, random값 만큼 카드의 Value에 더해주어 선택되는 카드가 게임을 실행할 때마다 다르게 해주었다. 또한, 카드 값의 위치 또한 무작위로 정해진다.

- Time: 시간은 카드의 수 \* 5로 Level별 시간이 다르도록 설정했다. 1초씩 줄어드는 효과를 추가하였으며, 시간이 0초라면, 시간초과로 Result Scene으로 이동한다.

- Score: 점수는 짝을 맞출 때 마다 각 Level별 카드의 수만큼 점수를 얻도록 설정했다. (4, 8, 18) 이때 얻을 수 있는 최대 점수는 202점이며, 시간 초과 또는 모든 Level을 마쳤을 경우, currentscore.txt파일에 점수를 쓴 후, Result Scene으로 전환한다.

- Matches: 현재 Level에서 몇 번의 짝을 더 맞추어야 하는지 나타낸다.

- Back: MenuBehavior Script를 사용하여 3을 전달한다. (Menu로 이동)

3. Result

- score: currentscore.txt파일을 읽어와서 ScoreText에 점수를 표시한다.

- Nick Name: 문자열을 입력하고 Submit 버튼을 누르면, Score와 현재시간과 함께 gamehistory.txt파일로 값을 저장한다. 또, Score와 Nick Name을 bestscore.txt파일로 값을 저장한다.

- Comment: 0점이하거나 202점을 넘는 경우: You’re a hacker.

0~49: TRY AGAIN!

50~99: CHEER UP!

100~149: NOT BAD!

150~201: GREAT!

202: PERFECT! 각 점수별로 격려하는 문구가 다르다.

- Back: MenuBehavior Script를 사용하여 3을 전달한다. (Menu로 이동)

4. Best Score

- Best Score: Best Score 오브젝트에 bestscore.txt의 모든 라인을 배열형식으로 입력 받은 후, 버블 정렬을 이용하여, Score가 높은 순서대로 배열을 내림차순 정렬한다. Score 값만 추출하기 위해 구분자로 ‘,’를 사용했으며, 정렬을 마친 후, 1위부터 마지막까지, 순서대로 출력한다.

문자열은 Score와 Nick Name으로 이루어져 있다.

- Prev: 앞 순위의 문자열을 출력한다. 단, 1위의 값을 출력하고 있을 경우, 넘어가지 못하도록 한다. (배열의 범위를 벗어나기 때문)

- Next: 뒤 순위의 문자열을 출력한다. 단, 마지막 순위의 값을 출력하고 있을 경우, 넘어가지 못하도록 한다. (배열의 범위를 벗어나기 때문)

- Back: MenuBehavior Script를 사용하여 3을 전달한다. (Menu로 이동)

5. Game History

- Game History: gamehistory.txt의 모든 라인을 배열형식으로 입력 받은 후 순서대로 출력한다. (WriteLine은 문서의 가장 마지막 줄에 출력함.)

- Prev: 앞 순서의 문자열을 출력한다. 단, 가장 앞의 값을 출력하고 있을 경우, 넘어가지 못하도록 한다. (배열의 범위를 벗어나기 때문)

- Next: 뒤 순위의 문자열을 출력한다. 단, 가장 마지막 값을 출력하고 있을 경우, 넘어가지 못하도록 한다. (배열의 범위를 벗어나기 때문)

- Back: MenuBehavior Script를 사용하여 3을 전달한다. (Menu로 이동)

Script별 설명

MenuBehavior

- triggerMenuBehavior함수

오브젝트별 클릭 시 값을 int형 값을 받아와 Scene 전환을 하는 함수이다.

0: Level1 Scene으로 이동

1: Best Score Scene으로 이동

2: 프로그램 종료

3: Menu Scene으로 이동

기타: Game History Scene으로 이동

Card

- Start함수

상태를 1로 설정하고 Manager 태그를 불러온다.

- setupGraphics 함수

카드들의 앞 뒷면의 이미지를 설정한다.

- flipCard 함수

상태 값과 실행 여부에 따라 상태와 이미지를 설정한다.

- cardValue 함수

카드의 Value값을 설정한다.

- state 함수

카드의 상태를 설정한다.

- initialized 함수

카드의 초기화 유무를 설정한다.

- falseCheck 함수

화면의 변화를 체크한다.

- pause 함수

상태 값에 따라 카드의 앞면을 출력할지 뒷면을 출력할지 정한다.

GameManager

- Awake 함수

Scene이 시작되면 점수를 표시한다. (단 Level 1일 때는 0을 표시한다.)

- Start 함수

Scene이 시작되면 시간을 정한다. (각 Level별로 카드의 개수 \* 5만큼)

이후 처음 설정된 시간을 Scene에 출력한다.

- TimerTake 함수

시간을 나타내는 변수 값을 1초 단위로 감소시키며 출력한다.

- Update함수

시간이 남았다면 TimerTake함수를 실행시킨다.

만약 시간이 다 되었다면, currentscore.txt파일에 점수를 입력한다.

또, 만약 카드 더미의 초기화가 필요하다면 초기화를 시키며, 카드가 눌릴 때마다, checkCard함수를 실행한다.

- initializeCards 함수

52개의 카드들 중 무작위로 선택하여 짝을 이루게끔 카드의 속성들을 정해준다. 이후 카드에 그래픽을 추가해준다.

함수가 종료될 때 초기화 상태를 나타내는 변수를 true로 설정한다.

- getCardBack함수

카드의 뒷면 값을 반환한다.

- getCardFace 함수

카드의 앞면 값을 반환한다.

- checkCards 함수

카드가 두개 선택되어 있다면, 두 개의 카드를 비교하는 함수를 호출한다.

- cardComparison 함수

두 카드의 값이 같다면, 남은 짝의 값을 -1 시키고, 점수를 Level별 카드의 개수에 비례하여 증가시킨다. 만약 남은 짝이 없을 경우 Level 1 -> 2, Level2 -> Level3, Level3 -> Result Scene으로 이동한다. 만약 Level3에서 Result로 이동하는 단계라면 currentscore에 점수를 입력하고 Result Scene으로 이동한다. 또, 예외처리로 score값을 초기화하고, Menu Scene으로 이동한다.

ScoreManager

- Start 함수

Scene이 시작하면 currenscore.txt의 마지막 줄을 읽어와 게임 결과를 출력한다. 또한 점수에 따라 격려문을 다르게 출력한다.

(Result Scene 설명 참고)

- Input 함수

Nick Name을 입력 받은 후 시간, 점수, 이름순으로 gamehistory.txt파일에 입력한다. 또한 점수, 이름순으로 bestscore.txt파일에 입력한다.

LeaderBoardManager  
- Start 함수

bestscore.txt파일의 라인의 길이를 받은 후 길이가 0이면 “No Data!”를 출력한다. 만약 길이가 1이라면 하나만을 출력하며, 2이상일경우 버블 정렬을 활용하여 점수가 높은 순서대로 내림차순 정렬한다.

- Update 함수

길이가 0이면 “No Data!”를 출력한다. 만약 길이가 1이라면 하나만을 출력하며, 2이상일경우 rank변수 값의 번지수의 str값을 출력한다.

- next 함수

Prev버튼을 클릭하면 rank값을 1 감소시킨다. (단, Rank가 0일 경우 감소시키지 않는다.)

Next버튼을 클릭하면 rank값을 1 증가 시킨다. (단, Rank가 (txt파일의 길이-1) 일 경우 증가시키지 않는다.)

HistoryManager

- Start 함수

bestscore.txt파일의 라인의 길이를 받은 후 길이가 0이면 “No Data!”를 출력한다. 이외에는 순서대로 i값에 따라 출력한다.

- Update 함수

길이가 0이면 “No Data!”를 출력한다. 이외에는 i값에 따라 변경한다.

- next 함수

Prev버튼을 클릭하면 i--를 한다. (단, i가 0일 경우 감소시키지 않는다.)

Next버튼을 클릭하면 i++를 한다. (단, i가 (txt파일의 길이-1) 일 경우 증가시키지 않는다.)

2. Implementation

Required implementation

- 총 52장의 카드 사용: 문양 4종류 \* 숫자 및 영어 13 종류 총 52장으로 이루어진 카드를 사용함.

- 랜덤으로 제공된 카드 중 동일한 카드를 연속으로 touch한 경우 뒤집히며, 점수가 올라감: 구현함.

- Stage 1: 2X2, Stage 2: 2x4, Stage 3: 3X6: 동일한 UI 구현함.

- 모든 Stage를 끝내거나, Timeout인 경우 게임을 종료하고 점수를 중앙에 보여줌. :Result Scene으로 전환되며 구현함.

- 각 action과 effect에 적절한 사운드 사용: 구현하지 않음.

(action의 경우 카드의 앞 뒷면이 바뀌는 정도로 구현됨)

Free implementation

- Stage 난이도에 따른 timer, score차별화: 카드의 개수에 따라 점수와 시간이 달라진다.

- Effect 구현 (Particle, Shader 등): 구현하지 않음.

- Best score 저장: bestscore.txt파일에 저장되며, Best Score Scene에서 확인가능.

- Timer animation: Level별로 1초씩 감소되는 애니메이션이 존재하는 것을 확인할 수 있음.

- Replay: Back버튼을 구현하여, Menu로 돌아가 다시 게임을 진행할 수 있도록 함.

- Game History: gamehistory.txt파일에 시간, 점수, 이름순으로 저장하였으며, Game History Scene에서 확인 가능.

- 기타 본인의 기획사항 적용

1. Level Scene matches: Level별 몇 번의 짝을 맞추어야 다음 단계로 넘어가는지 화면 하단에 표시함.

2. Result Scene Comment: 획득한 점수 별로 격려문구가 달라지도록 구현함.

3. Best Score Scene Leader Board: Best Score Scene에서 버블 정렬을 활용하여, 점수가 높은 순서대로 순위를 확인할 수 있도록 함.

4. 점수 저장에 Nick Name 추가: 어떤 사용자의 점수인지 확인하기 쉽도록 사용자에게 문자열을 입력 받아 저장함.

3. Consideration

문제점

- 같은 카드를 두 번 클릭할 경우 다른 카드를 선택할 수 있음.

(통상적인 규칙으로 한 번 뒤집은 카드는 뒤집을 수 없어야 하지만, 게임 구현이 정상적으로 되었는지 확인하기 용이하도록 구현함.)

- Effect와 Sound가 구현되지 않음.

(실제 사용자의 경우 소리나 효과에 따라 프로그램 사용의 만족도에 큰 영향을 주지만, 게임 구현에 집중해 효과를 넣지 못했음.)

- 카드의 Image 로딩 시간이 김.

보완할 점

- Best Score Scene이나, Game History Scene에서 Text 오브젝트를 추가해 한 번에 더 많은 정보를 읽을 수 있게 할 수 있음.

- Effect와 Sound 추가

- 카드의 Image 로딩 시간 개선: 게임을 진행할 경우, 초반 카드 더미 구성에 시간이 오래 걸림 (사진이 너무 크거나, 코드의 효율이 떨어진다는 것을 의미) 그렇기 때문에 Level을 넘어가기 전 지연시간을 두고 로딩창을 만든다면 실제 플레이 시간에 영향이 없도록 할 수 있음.

-게임을 정상적으로 돌린 이후 데이터를 한 번 이상 넣어야만, Best score Scene과 Game History Scene이 정상적으로 돌아가므로, file이 없을 때 클릭해도 Scene 전환이 이루어지지 않도록 수정한다.

기타

- 파일 입출력의 경우 운영체제의 종속적인 부분이라서 윈도우에서 파일 입출력이랑 안드로이드 파일 입출력이 아예 개념 자체가 다를 수도 있음

- Resources폴더 내에 파일이 존재할 경우 에디터상에서는 write가 되지만, 빌드할 경우 읽기 전용이 된다는 견해가 있다.

- rwx = Read Write eXecute

-안드로이드 환경에서의 파일 입출력을 위해 Application.persistentDataPath를 사용하였다. (안드로이드 내부저장소 파일 읽기 쓰기 가능)

- 안드로이드 운영체제를 사용하기 위해서 LDPlayer4를 사용함.

4. Reference

<https://doggie-development.tistory.com/3>

<https://ssabi.tistory.com/12>

<https://3dmpengines.tistory.com/1745>

<https://www.youtube.com/watch?v=qrIbnvSTVs8>

<https://devilchen.tistory.com/4705>

<https://docs.unity3d.com/kr/2018.4/Manual/class-TextAsset.html>

<https://moblieandlife.tistory.com/entry/%ED%8C%8C%EC%9D%BC-Unity-Resource%ED%8F%B4%EB%8D%94%EC%9D%98-%EB%B0%94%EC%9D%B4%EB%84%88%EB%A6%AC%ED%8C%8C%EC%9D%BC-%EC%9D%BD%EA%B8%B0>

<https://m.blog.naver.com/PostView.nhn?blogId=clater11&logNo=80132855328&proxyReferer=https:%2F%2Fwww.google.com%2F>

<https://coderzero.tistory.com/entry/%EC%9C%A0%EB%8B%88%ED%8B%B0-%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8-%EC%86%8C%EC%8A%A4-%ED%85%8D%EC%8A%A4%ED%8A%B8txt-%ED%8C%8C%EC%9D%BC-%EC%9D%BD%EA%B3%A0-%EC%93%B0%EA%B8%B0>

<https://drehzr.tistory.com/489>

<https://worldeye.tistory.com/entry/Unity-%ED%85%8D%EC%8A%A4%ED%8A%B8-%ED%8C%8C%EC%9D%BC%EC%9E%85%EC%B6%9C%EB%A0%A5>

<https://mentum.tistory.com/157>

<https://076923.github.io/posts/C-22/>

<https://hyunity3d.tistory.com/102>

<https://docs.unity3d.com/kr/2019.4/Manual/android-BuildProcess.html>

<http://blog.naver.com/PostView.nhn?blogId=dktmrorl&logNo=220405012763>

<https://www.youtube.com/watch?v=fCGdaQ50DbA&list=PLbCx65TBvT-RHOTE_Lt7iTAev-4Bf84Fu>

<https://www.youtube.com/watch?v=qaCjBh7bWz0&list=PLZhNP5qJ2IA2DA4bzDyxFMs8yogVQSrjW>

기타 등등…