|  |
| --- |
| **Cragene Rabbit** |

**<최종 결과 보고서>**

학과 : 게임공학과

학번 : 2012180004

이름 : 권창현

게임소개

1. 게임 이름

* Crazy(미친)Gene(유전자)Rabbit(토끼)를 미친의 뜻을 가진 크레이지와 유전자의 뜻을 가진 진을 합쳐서 Cragene Rabbit으로 이름을 정하였다

1. 게임 방식

* 종 스크롤 방식의 점프 게임
* 좌.우 키보드를 이용하여, 장애물을 밟고 올라간다.

1. 개발 환경

* Windows 기반 (480x750)
* WinAPI 이용

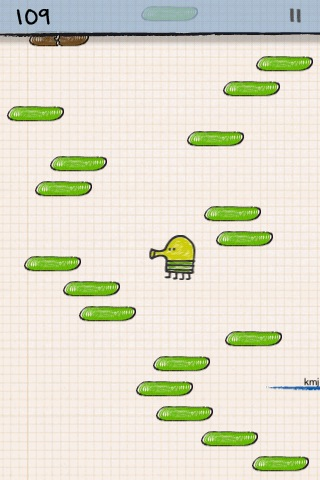
1. Game Story

2033년 과학발전으로 인하여 유전자를 조작 할 수 있게 되면서, 토끼를 실험대상을 선정하여 유전자 조작으로 토끼를 태어나게 하였다. 조작된 유전자는 기존 토끼보다 수십 수백 배 이상의 점프 능력과, 다 지능적인 사고능력을 갖게 되어, 토끼는 과학자 들로부터 도망을 치기 위하여 계속해서 점프를 하여 하늘 위로 도망을 치고 있다. 이를 알아차린 과학자들이 토끼를 잡기 위하여 많은 실험 도구를 이용하여 토끼를 잡으려고 한다..!

1. 예시 게임

* 리마 스카이 – **“Doodle Jump”**

게임 화면 & 진행부분



1. 게임의 특징

사용자가 따로 게임에 대한 기술이나 공략 등을 익히지 않아도 누구나 쉽게 게임을 즐길 수 있으며, 점수와 같은 판별 할 수 있는 기준이 있으므로 친구들과의 경쟁을 유도하여, 게임의 집중과, 게임이용시간 등을 증가시킬 수 있는 장점을 가지고 있다.

1. 실행 방법

프로그램 실행 시 PLAY, Multi-Play, Exit, Help 등의 메뉴가 나타난다.

버튼을 마우스 커서를 가까이 되면 회색으로 변화게 되어있으며, PLAY 버튼 외 나머지 3개의 버튼 모두 마우스 커서를 가까이 되면 회색으로 변화게 되어있다.

각각의 앞 글자를 이용하여 마우스 클릭대신 PLAY면 P를 누르면 해당 버튼이 자동으로 눌러지는 단축키 효과를 집어 넣었고, 게임이 시작된 후 에는 키보드의 좌우버튼을 이용하여, 캐릭터를 움직여 구름을 밟을 수 있다.

캐릭터의 점프는 따로 직접 점프를 할 수 있는 키를 누르거나, 하는 것 이 아닌 캐릭터가 스스로 점프를 할 수 있게 만들어져 있다.

1. 필요한 라이브러리

User32.Lib

* 게임의 처음 시작과 메뉴간의 화면 전환, 게임 도중의 검정색의 구름 등에 사용되어지는 라이브러리로, AlphaBlend에서의 특정 비트맵 이미지만 투명하게 하는 것이 아닌 프로그램 전체의 투명도를 제어할 수 있으며, User32.Lib 라이브러리 SetLayeredWindowAttributes 함수를 사용하여 프로그램의 투명도를 조절하였다.

Winmm.lib

* 게임에 진행에 있어서 효과음 등의 사운드를 사용하기 위한 라이브러리 이다. 해당 라이브러리를 사용하기 위해서 “mmsystem.h”를 사용하였으며, Winmm.lib 라이브러리 PlaySound 함수를 사용하여 프로그램의 효과음을 사용하였다.

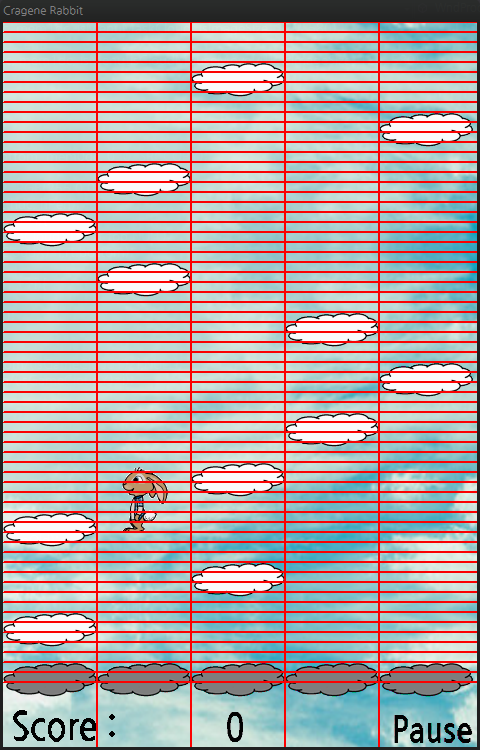
1. 구현한 기술 & 스크린샷



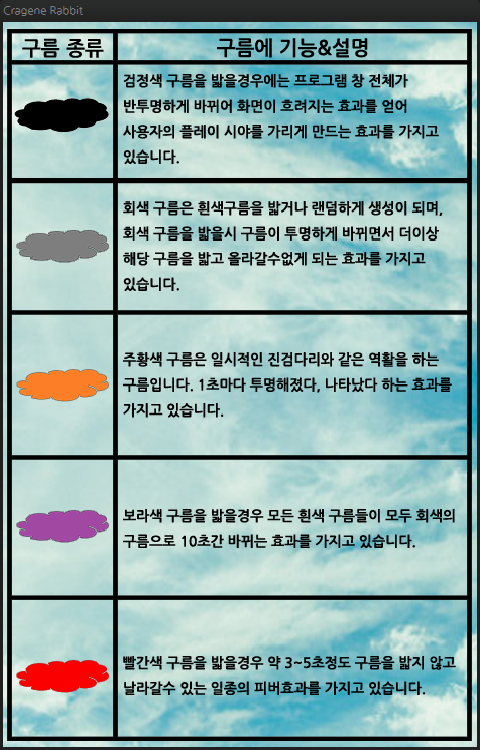
PLAY, Multi-Play, Exit, Help 등 프로그램 내부에 있는 모든 이미지는 **마스크 이미지**를 사용을 하였다.

처음 구현할 당시의 이미지를 표시 해 줄 때에는 **TransparentBlt**를 사용하여 이미지를 표시 해주었지만, 많은 이미지와 저사양 컴퓨터에서의 동작의 느림을 확인하여 **StretchBlt**를 사용한 마스크 이미지를 사용하였으나, 이 또한 저사양 컴퓨터 등에서의 동작이 일부 느리거나 끊기는 현상이 나타나서 이미지 크기를 그려줄 크기만큼으로 조절하여 고속복사를 할 수 있는 **BitBlt**를 사용하여 저사양 컴퓨터에서도 끊기거나, 느려지는 현상을 최소화 하였다.

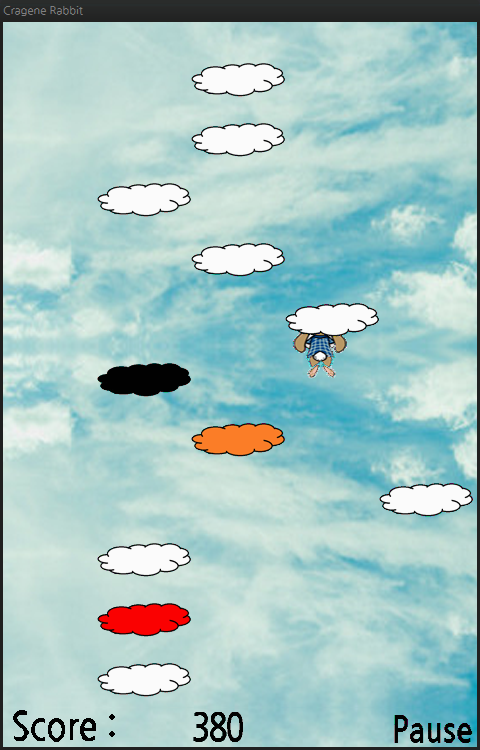
게임 도중 마우스가 다른 곳을 클릭하여 게임이 다른 화면 뒤로 가능 것을 방지 하기 위하여 **SetWindowPos 함수**를 사용하여 **HWND\_TOPMOST 최상위 윈도우**를 사용할 수 있게 구현을 하였다.

게임의 진행을 하면서 구름을 밟고 올라가는 부분과, 구름이 나오는 부분은 **[66][5] 크기의 배열**을 이용하여 구름의 생성과, 소멸, 등을 나타내었다.

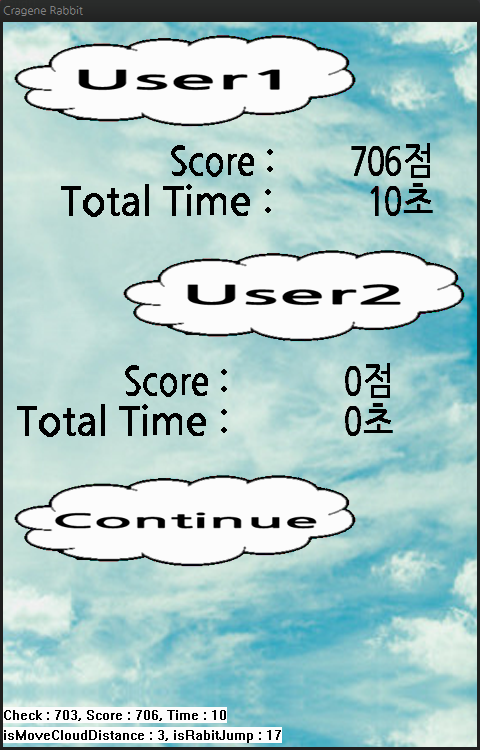
구름과 캐릭터의 충돌 체크는 구름이 생성되는 위치의 **RECT**값을 받은 후, 토끼(캐릭터)의 RECT값과 비교를 하였으며, 구름이 토끼(캐릭터)보다 위에 있지만 같은 위치에 있는 것을 막기 위하여 배열의 위치를 판단하여, **현재 구름이 있는 위치의 배열보다 4칸위의 배열 안에 있을 경우**에 해당 구름에 충돌체크를 할 수 있게 제한을 걸어 두었다.



각각의 구름마다의 적용되는 다르게 되어있는데 이는 배열에 데이터를 “**1=일반구름, 2=회색구름, 3=투명구름, 4=주황구름, 5=빨강구름, 6=보라구름, 7=검정구름**”과 같이 정해주어 그림을 그려줄 때 이미지를 교체하는 방식과 충돌체크 후 어떤 구름과의 충돌을 했는지 확인을 하여 해당 기능에 맞게 기능을 실행하게 만들었다.

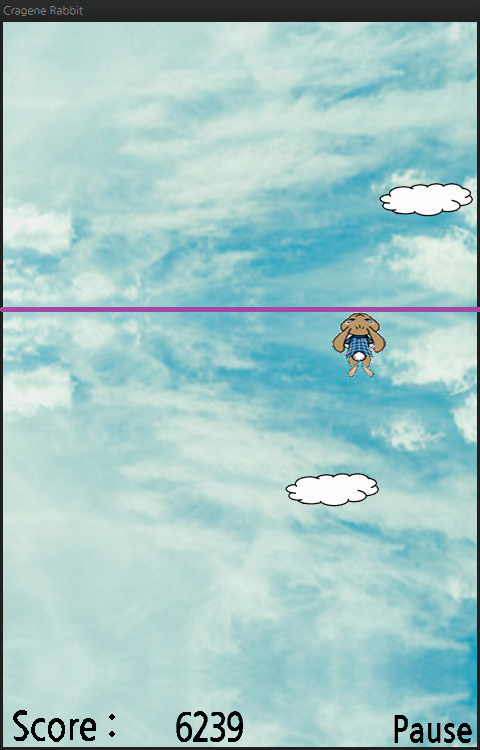
구름의 경우 처음에는 일정한 패턴을 만들어서 구름을 띄우려고 하였으나, 그렇게 될 경우 반복되는 패턴으로 인한 게임의 난이도가 쉬워지므로, **isRandCloud()** 함수를 만들어서 rand()%5를 하여 랜덤 한 위치에 구름의 데이터를 넣어주게 만들었다. 처음 랜덤 하게 만들어 주려고 **srand((unsigned int)time(NULL));** 을 사용하여 시드 값을 바꾸어 주었는데, 빠르게 구름들이 내려가다 보니 시드가 변경되지 않아서, **srand(isCheck%**

**GetTickCount()+rand()%isCheck);** 와 같이 계속해서 값이 바뀌는 점수, 체크, 시스템 시간을 섞어 주어 시드가 변경되게 만들었다.

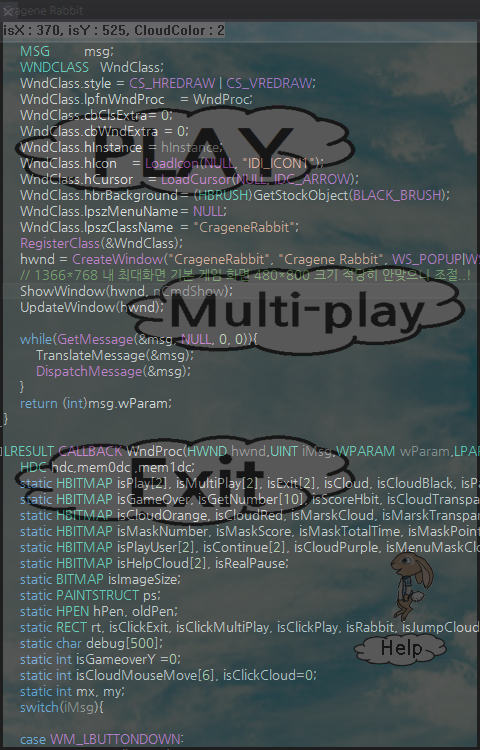


멀티플레이 같은 경우에는 동시에 게임을 진행하는 방식이 아닌 각각의 사용자가 한번씩 게임을 하여금 왼쪽 사진과 같이 점수와 시간을 비교하여 누가 더 높은 점수를 받았는지 대결할 수 있는 게임의 진행방식이다.

TotalTime 이라는 시간을 구하는 부분은 **“Windows.h”**에 포함되어있는 **GetTickCount()** 함수를 이용하여 게임 시작 시 불러온 후, 게임이 끝난 뒤 함수를 불러와 두 값의 차를 뺀 나누기 1000을 하여 지나간 시간을 구하였다. **GetTickCount()**에서의 시간은 **ms 단위**의 시간을 기록하기 때문에 **ms 단위**에서 **s 단위**로의 변환은 1024를 나눠야 하지만 편의상 1000으로 나누어 계산을 하였다.

게임을 플레이 하면서 어느 정도 시간이 점수가 올라가면 왼쪽의 사진과 같이 구름의 거리가 확연하게 많이 띄어져 있는 차이를 볼 수 있다. 처음 시작 할 때 와는 확연하게 거리 차이가 있는데, 게임이 진행되면서 **토끼(캐릭터)의 점프 높이와, 구름의 거리가 증가**하면서 저러한 길이가 나오게 된다.

토끼(캐릭터)는 왼쪽에 그려져 있는 **보라색 선 이상을 넘어갈 수 없게 구현**을 해놓았다. 처음에는 선이 없이 점프하면 점프한 만큼 올라가게 설정을 하였더니, 토끼(캐릭터)가 항상 제일 위에서 다리만 보이면서 점프를 하고 있어서 위와 같이 제한선을 두어 제한선을 넘어가면, 구름이 내려오는 방식을 사용하게 되었다.



게임의 화면 전환 및 게임 도중 검정색 구름을 밟을 경우 왼쪽 사진과 같이 프로그램이 투명해 지면서 뒷배경이 보이는 부분이 있다. **User32.lib 라이브러리**를 이용하여 폼 전체를 반투명 하게 만드는**SetLayeredWindow**

**Attributes 함수**를 사용 하였다. 해당 함수는 **AlphaBlend** 와는 별개로 폼 전체를 반투명하게 만들 수 있으며, **User32.lib 라이브러리**에는 위와 같은 함수 말고도 **윈도우 관련한 제어** 등을 할 수 있는 많은 함수를 내장하고 있다.

1. 제작 후기

두들점프 라는 게임을 처음 시작 하였을 때가 고등학교 1학년~2학년 시기에 안드로이드가 막 보급화 되기 이전에 나온 게임이다. 그때 당시에는 두들점프 게임을 하면서, 게임 정말 못 만들었다. 재미없다 등등… 별 애 별 생각을 하면서 게임을 하였는데, 그런 게임을 막상 내가 직접 만들려고 하니깐 정말 간단하게 게임이 구현이 되는 게 아니 구나 라는 것을 느끼게 되었다.

또한 계속 게임을 제작하면서 느끼게 된 것이, 1학년 수업, 2학년 수업을 들으면서 프로그래밍 수업만 들으면 되지, 왜 수학, 물리 등의 수업은 왜 듣는 건지 이해가 가지 않았는데. 게임을 제작을 하면서 프로그래밍을 하는 것 보다 수학이나, 물리 와 같은 공식을 사용하여 프로그래밍을 하는 것이 훨씬 더 많은 것을 느끼게 되었다. 물론 최대한 수학적인 부분이나, 물리와 같은 어려운 부분을 프로그래밍으로 안 하기 위해서 최대한 쉽게 프로그래밍을 하였지만, 앞으로의 프로그래밍을 하려면 적어도 수학, 물리는 필수적으로 공부를 해두고선 프로그래밍을 들어가야 한다는 것을 느끼게 되었다.

끝으로 지금까지 윈도우 프로그래밍을 했던 프로젝트를 보면서 지금의 게임을 제작하게 되었는데, 처음 윈도우 프로그래밍을 했던 프로젝트와 지금의 프로젝트를 보면 정말 확연하게 차이가 많이 나는 것을 느끼게 되었다. 처음 프로젝트에는 책에서 그대로 똑같이 타이핑을 한 느낌이 강하게 드는 반면에 지금 내가 만든 게임의 프로젝트는 나만의 코딩 방법과 같은 내가 자주 사용하는 방식들이 들어간 것을 보면서 신기함을 느끼게 되었다.

처음에 윈도우프로그래밍을 수강 할 때에는 이런 간단한 걸로 어떻게 게임을 만들어? 하면서 생각을 하였는데, 지금 완성을 한 게임의 코드를 보면 대부분이 간단한 코드들이 합쳐져서 하나의 큰 코드로 이루어진 것을 보면서 정말로 신기함이 저절로 나올 수 밖에 없었다.

끝으로 앞으로는 게임 개발자들에게 이런 재미없는 게임을 왜 만들었냐는 둥에 말을 다시는 하지 말아야겠다. 내가 생각해도 게임을 만든다는 것 하나 만으로도 정말 대단하고 존경스러운 개발자라는 것을 알게 되었다.