DreamCoast Portfolio Project

DREAMCOAST2D

2D Portfolio 기획 및 기술서

Heedong "Arkiny" Lee

Version	Date	Author	Change
091314	14/09/13	HD	Initial Document / The First Proposal
091514	14/09/15	HD	미팅결과 반영, Framework Initial Design
091614	14/09/16	HD	개발 시 문제점 발견, 추가, Diagram Update
092414	14/09/24	HD	Diagram, Challenge Update
093014	14/09/30	HD	Diagram Update
100614	14/10/06	HD	Diagram Update, Challenge Update
110314	14/11/03	HD	Diagram, Module Interface, Diary Update

목 차

1 서론

1.1 본 문서의 목적

본 문서는 차후 해당 프로젝트를 이해하려는 자를 돕기 위해 작성한다.

또한 Project Manager 의 참고 자료로도 쓰일 수 있다.

1.2 본 문서의 적용 범위

포트폴리오에 적용된 모든 범위에 해당하며,
코드 내 각주의 설명이 부족해 추가적인 첨언이 필요할 경우, 본 문서에 적용한다.

1.3 정의, 유의어, 약어 목록

단어	설명
D2D	DirectX 2D 의 약어
IrrKlang	사용된 사운드 엔진

1.4 디자인 목표

1. 내가 플레이 할 때 재미있는 게임 제작
2. 3D 과정에 들어가기 전, 2D 게임제작을 통해,
   C, C++, STL, WinApi, D2D 숙달 정도를 자가 평가 하기 위한 포트폴리오 제작
3. 포트폴리오 제작 과정을 통해 일반 전산학에서 생각해 볼 수 없었던 게임 개발 실무 과정에서의 프로그래밍 기법 숙달

2 REFERENCES

2.1 MSDN

- http://msdn.microsoft.com/en-US/
- C++ 및 WinApi 용법 참고

2.2 CPLUSPLUS REFERENCE

- http://www.cplusplus.com/
- C/C++ 용법 참고

2.3 DIRECT2D REFERENCE

- http://msdn.microsoft.com/en-us/library/windows/desktop/dd370990(v=vs.85).aspx
- Direct2D 용법 및, 사용 예시 참조

2.4 DIRECTSOUND REFERENCE

- http://msdn.microsoft.com/en-us/library/windows/desktop/ee416960(v=vs.85).aspx
- DirectSound 용법 및, 사용 예시 참조

2.5 IRRKLANG SOUND ENGINE

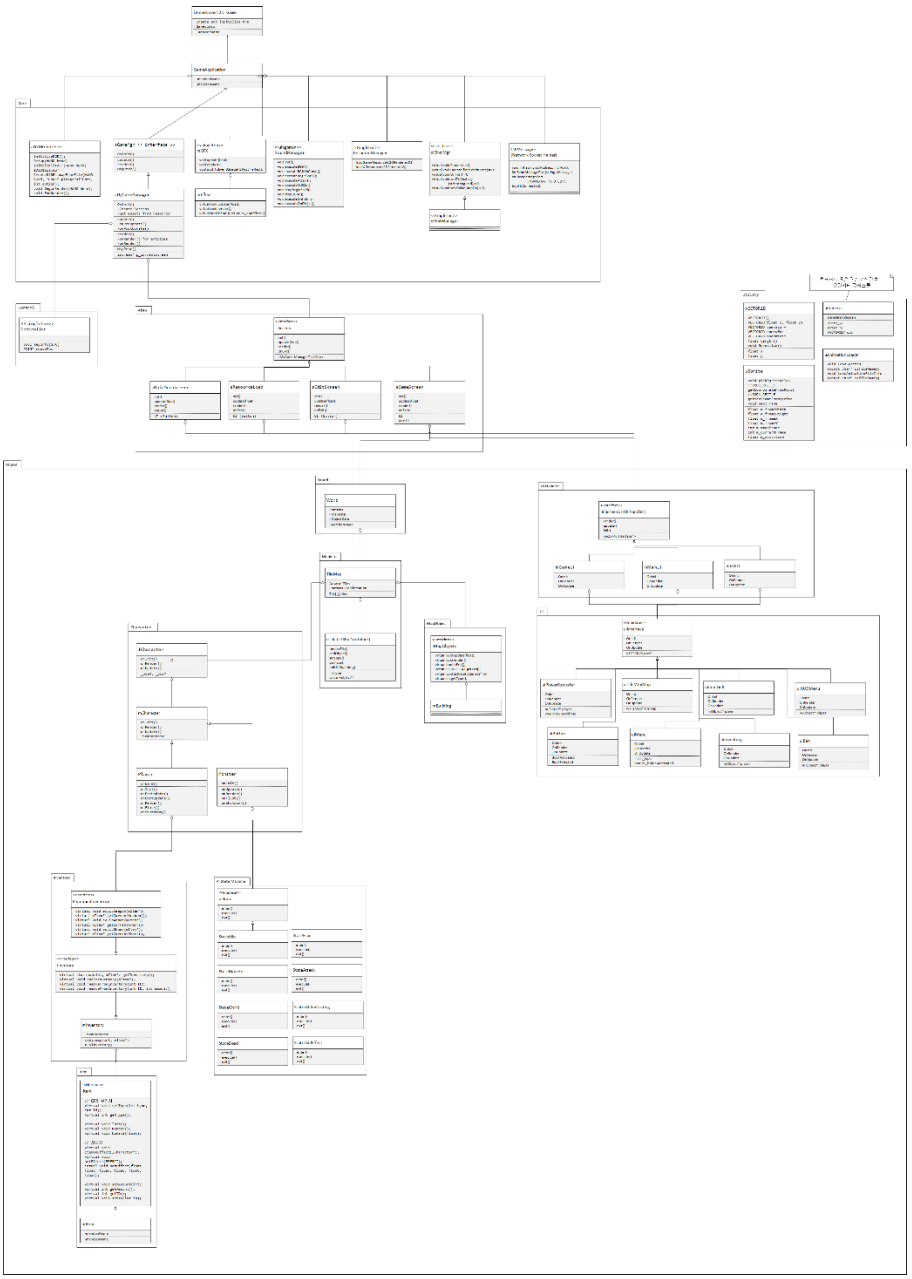
- http://www.ambiera.com/irrklang/
- Irrklang sound 엔진 용법, 사용 예시 참조

중심적으로 참고한 곳만 기재하였으며, 각 세부 별 기술을 참고하였을 시,
해당 항목에 각주로 기재

3 프로그램 세부 설명

3.1 각 모듈 설명

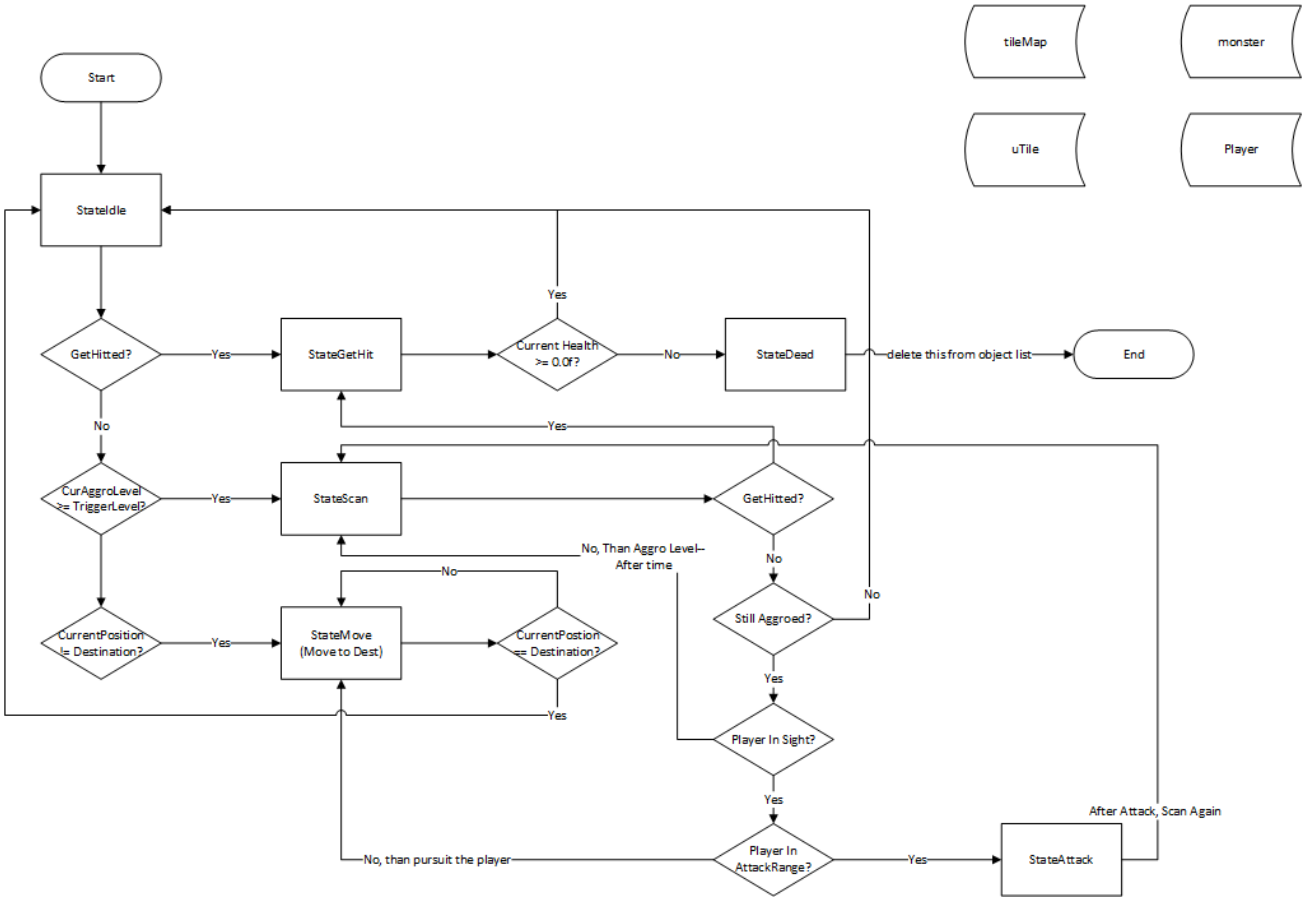
3.1.1 Class Diagram



- 가장 기초적인 MVC 모델[1]을 이용, Basic Framework 작성 (visio 및 그림파일 첨부)

[1] Model, View, Control : http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

3.1.2 AI State Flowchart (Passive)



3.2 동기화 과정 설명

- 기본적으로 초기화, 업데이트, 렌더링 순으로 이루어진다.
- 현재 게임 자체는 싱글쓰레드로 실행되므로 데이터 동기화 핸들링은 크게 필요가 없다.
- 채팅 서버 부분에서 현재 쓰레드를 이용하고 있으나, 데이터를 저장하지 않으며,
  - ■ 출력 쓰레드
  - ■ 입력 쓰레드
- 로 이분화 되어 이루어지기 때문에 동기화에서 ThreadSafe 는 어느정도 이루어진 상태

4 의존성 설명

4.1 모듈간 의존성

- mPlayer 클래스는 wTileMap 의 포인터를 받아와 맵정보를 판단해 이동불가 처리를 하고,
- wTileMap 클래스는 mPlayer 클래스 포인터를 받아와 렌더를 하므로, 상호 데이터 의존성이 있다. 따라서 둘 중에 하나의 정보가 바뀔 경우, (new 를 통해 새로운 정보를 할당했을 경우) 양쪽의 포인터를 갱신 처리해줘야 한다.
- 초기 개발 단계에서 각 모듈간 직접 포인터 통신을 위해 포인터로 각 클래스간 통신하는 경우(상기 사항 참조)가 있으므로, 각 모듈간의 종속과 의존성이 강하다.

4.2 프로세스간 의존성

- UserInteface 를 상속받는 클래스중 일부 클래스(미니맵, 스탯바, 아이템 벨트)의 경우, 월드 맵이 가지고 있는 클래스들(타일맵, 플레이어 등)을 의존한다.
- 이외에도 전체적인 큰 Loop 안에서 어플리케이션이 실행되므로, 초기화 순서 등의 의존성에서 자유로울 수가 없다.

4.3 데이터간 의존성

- mPlayer 내부의 _realPos 변수와 _drawPos 는 유기적으로 상호작용을(_drawPos 가 _realPos 를 참조하여 최초 위치 설정후 동시 이동)하며, 이동불가 처리 및 스프라이트 피봇을 결정한다.
- InGameUI 는 World 클래스가 가지고 있는 Player 클래스와 TileMap 클래스의 정보에 의존해 UI 를 렌더링한다.
- 역시 큰 Loop 안에서 어플리케이션이 실행되므로, 각 클래스 들이 참조하는 데이터간 업데이트 정보에 대한 의존성에 유의하여야 한다.

5 인터페이스 설명

5.1 모듈 인터페이스 설명

상속시 최상위 베이스 클래스가 아닌,
실질적으로 _interface 키워드를 통해 인터페이스를 적용한 클래스 중심으로 기술.

5.1.1 cIGameMgr

- 게임 매니저의 최상위 인터페이스
- 실질적 게임의 초기화, 업데이트, 렌더 순환의 가장 상위에 위치한다.
- 이 매니저가 콜백되어 씬전환이 이루어진다.

5.1.2 cIChatMgr

- 채팅 네트워크 매니저의 최상위 인터페이스
- 네트워크 유저인터페이스의 출력을 담당한다.

5.1.3 sIScreen

- 게임내 스크린의 가장 최상위 인터페이스, 게임의 씬 전환이 이 인터페이스 간에 이루어진다.

5.1.4 iInScreenUI

- 스크린 내 UI 를 총괄하는 클래스, UI 매니저 역할을 한다.

5.1.5 uiInterface

- 유저 인터페이스를 담당하는 클래스

5.1.6 iMapObject

- 맵에 출력될 오브젝트 모듈 인터페이스

5.1.7 ICharacter

- 상기된 오브젝트를 제외한 캐릭터들에 관련된 모듈 인터페이스

5.1.8 aiState

- 몬스터의 AI 모듈 인터페이스

5.1.9 IInventory

- 인벤토리 관리에 쓰이는 모듈 인터페이스

5.2 세부사항 설명(각주 외에 필요한 경우)

- 현재까진 불필요

6 위와 같이 디자인한 이유

6.1 개발(디자인) 중 어려웠던 점 및 해결책

6.1.1 증감연산 속도 차이

- 문제발견: 캐릭터의 이동을 구현하던 중 발견한 사항으로 단순한 플러스 마이너스 이동으로는 증가 이동(우측 혹은 하향이동), 감소 이동(상향, 혹은 좌측이동)간의 연산차이가 존재함을 발견.
- 문제 해결 가설
  - ■ 현재 문제는 Scalar Vector 를 이용하고 있기 때문에 일어나는 것 일수도 있다.
  - ■ Direct2D 는 Direct3D 의 연장선이므로, 그 영향이 있을 것이다라고 추론

  ➔ 따라서 Point 를 이용하는 대신 VECTOR 클래스를 이용하여 이동을 구현하기로 결정
- 문제 해결
  - ■ 해당 문제는 POINT Structure 에 위치정보를 담았기 때문에 생기는 문제 (위치정보는 Float 으로 관리되고 있었다.)

  ➔ Float 을 이용해서 VECTOR2D 클래스를 만들어서 이동관리를 하게 되면 자동적으로 문제가 사라짐

6.1.2 Render Ordering

- 문제발견: 캐릭터의 위치가 타일 위에 고정되어 있는 방식이 아니므로, 아래로 이동 시 타일에 의해 캐릭터 스프라이트가 일정부분 가려지는 문제가 발생.
- 문제 해결 가설
  - ■ 가설 1. 피벗의 세심한 수정을 통한 수정
  - ■ 가설 2. 타일 렌더 -> 캐릭터 렌더 -> 오브젝트 렌더 순으로, 타일위의 오브젝트를 따로 렌더하는 방식으로 처리
- 문제 해결
  - ■ 가설 1 에 따라 피벗의 수정을 통해 대부분의 문제 해결
  - ■ 플레이어 캐릭터 좌표를 따로 설정함으로써 플레이어 캐릭터 위치 객관성 확보

6.1.3 Mob Render Ordering

- 문제발견: Object 의 렌더 역시 타일 위에 고정되어 있는 방식이 아니며, 역시 Sorting 이 되어야 하기 때문에, tile 이 렌더 될 때 동시에 렌더해야한다.
- 문제 해결 가설

- ■ 가설 1. 각 타일을 컨테이너화 하여, Update 시에 각 object 들을 좌표에 맞는 타일에 넣고, 렌더시에 해당 컨테이너를 비우는 과정 반복

- 문제 해결

  - ■ 가설 1 에 따라 해결

6.1.4 Mob Render Ordering (2)

- 문제발견: 6.1.3 항목의 방법대로 처리했으나, pivot 좌표와, 위치 좌표가 일치하지 않기에 타일에 스프라이트가 가려지는 문제점 발견

- 문제 해결 가설

  - ■ 가설 1. 전투용 컨테이너와, 렌더용 컨테이너를 나누어서 처리

- 문제 해결

  - ■ 일단 몬스터의 pivot 을 현재 위치로 치환해서 처리

6.1.5 User Interface

- 문제 발견: UI 를 하나의 게임내의 오브젝트로 취급할 것인가?
- 문제 해결 가설
  - ■ 가설 1. Object 로 처리
    - ◆ 오브젝트로 처리하면, 카메라 연산 등, 신경 쓸게 많아진다.
  - ■ 가설 2. 스크린 내에 하나의 레이어를 추가해 UI 를 게임 위에 깔아준다.
- 문제 해결
  - ■ 가설 2 에 따라 Screen 에 UI 레이어를 하나 더 깔아주는 식으로 처리
    - ◆ 장점: 한결 편해진 UI 관리, 카메라가 필요 없음.
    - ◆ 단점: 마우스 클릭 시 World 와 중복적용이 될 가능성이 있다.
      - ● 게임 내적 방법으로 처리(확실한 2 원화, 메뉴 활성화 시 월드 pause)

7 개발 마무리 후 이후 계획

(고려할 점 별로 차후 지원계획 추가)

8 부록

8.1 부록 1, 개발 자원

8.1.1 개발 인원

- 개발인원: 1 명
- 개발인원 상세

  이희동

  프로그래밍 경력

  C, C++, Java: Advanced Level

  C#, XML, MySQL: Intermediate Level

  학력 사항

  Bachelor of Science in Computer Science, Iowa State University (Jan 2010 ~ May 2014)

  경력 사항

  XAA Database Project Associate Programmer (May 2011 ~ Aug 2011)

8.1.2 포트폴리오 제작 기한

- (9 월 14 일 ~ 11 월 25 일) 휴일포함 약 70 일
- 시간자원 관리: Microsoft Project 2013 – Project1.mpp

8.1.3 버전관리

- GitHub : https://github.com/arkiny/DreamCoast2D

8.2 부록 2 최초 개발 제안

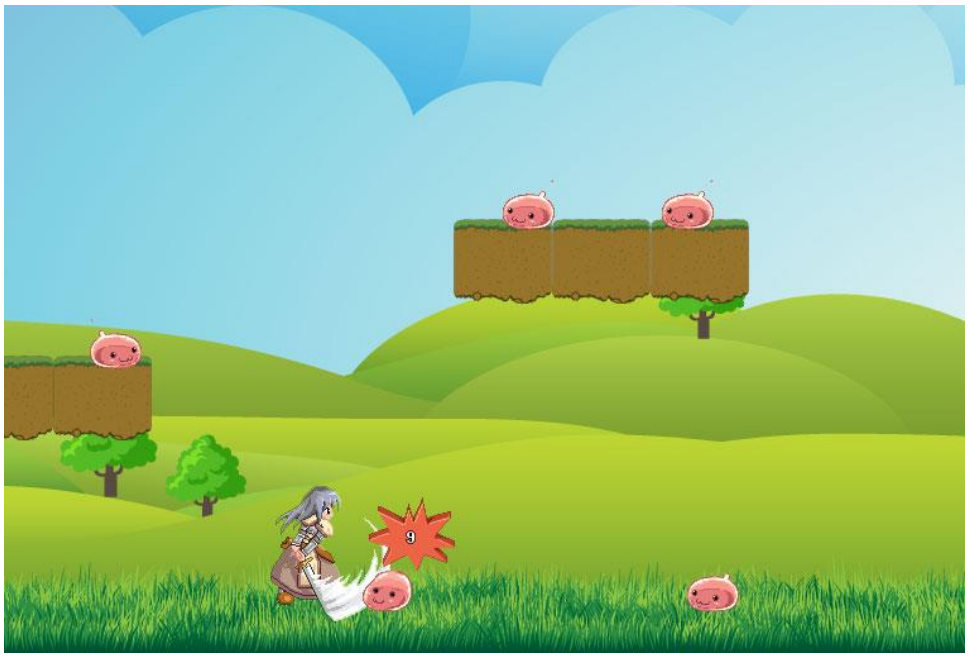
8.2.1 최초 아이디어

- 어렸을 때 재미있게 플레이 했고, 상대적으로 그래픽 리소스가 많이 공개되어 있는 Ragnarok Online 및 Ragnarok Battle Adventure 의 리소스[23]를 이용한 키보드 플레이형 Action Role Playing Game.

8.2.2 플레이 방식에 관련된 제안사항

8.2.2.1 플랫포머 방식

- 개발 모티브: 마리오, 메이플 스토리
- Concept Screen[45]



- 장점: 쉽다.
- 단점: 너무 무난하다.(?)

---

[2] 라그나로크 온라인 스프라이트 : http://rosprites.blogspot.kr/

[3] 라그나로크 배틀 어드벤쳐 : http://spritedatabase.net/game/1738

[4] Player Sprite - Ragnarok battle Adventure

[5] Monster Sprite - Ragnarok Online

8.2.2.2 2D 벨트스크롤 방식

- 개발 모티브: 오락실 던젼 앤 드래곤, 던젼 앤 파이터 온라인

- Concept Screen[67]

- 장점

  - 리소스를 구하기 쉽다.
  - 초기 단계에서는 무난하게 코딩 할 수 있다. (스프라이트 신경 쓸 게 적은 편)

- 단점

  - 결국 던파를 벗어나기가 힘들다. (게임이 식상하다)
  - 목표로 했던 방식(타일피격)을 적용하기에 약한 편이다.
  - (힛박스를 이용한 타격범위가 애매해진다-> 타격범위관련(힛박스)하여 생각할 게 많아지는 편이다.)
  - (공중 스킬은 지정할수 있겠지만, 공중에 떠있을때의 타격판정이 힘들어진다.) (x, y, z 3D 모든 축을 고려해야 한다)

---

6 Player Sprite - Ragnarok battle Adventure

7 Monster Sprite - Ragnarok Online

8.2.2.3 2.5D (y 축이 없는 액션형 RPG 게임)

- 개발 모티브: Tree of Savior, HammerWatch, 쯔바이

- Concept Screen[8]

- 장점

  - ■ 원래 라그나로크처럼 타일형식의 타격범위를 지정하기에 어색함이 없는 편이다.
  - ■ 타일 방식을 이용함으로 효율성과 생산성을 뽑아낼 수 있다.

- 단점

  - ■ 소유중인 리소스(스프라이트) 렌더링이 어려운 편이다.
    (확보할 수 있는 리소스가 대각선 공격 스프라이트밖에 없어서, 바로 위나 아래 공격 시 어색함이 묻어나올 수 밖에 없다.)
    (공격범위를 왼쪽 위를 볼 땐 (-1, -1) (-1 ,0) (0, -1) 모두 타격하게 처리?)

[8] Player, Monster Sprite - Ragnarok Online

8.2.3 구현 범위에 관한 제안 사항

8.2.3.1 재미 중심형

- 코딩하는 사람이 재밌으면 남들도 재밌어! 내가 플레이 중에 재밌게
- 장점: 내가 재밌을 듯.....
- 단점: 목적한바(포트폴리오용)이라는 목적에 비해서 너무 어정쩡할거 같다.

8.2.3.2 전체 완성형

- 스테이지 숫자를 추가하면서, 필요한 것만 효율적으로 집어넣어서 완성한다.
- 장점: 플레이 하는 사람은 재밌다.
- 단점: 역시나 오랜 코딩 시간, 내가 가진 모든 것을 표현할수 있을까?

8.2.3.3 기술 시연형

- 스테이지가 적고, 내가 할 수 있는걸 최대한 다 집어넣는다.
- 장점: 목적에는 부합하다.
- 단점: 플레이 타임이 한정되므로 실제 플레이 해보는 사람에게 있어서는 심심하다.
  할 수 잇는 것은 다 넣으므로 프로그램이 지저분해 질 수 있다.

8.2.4 구현 툴에 관한 제안사항

8.2.4.1 WinApi

- 장점: 범용성, 제작자에게 익숙함
- 단점: 차차 도태되가는 기술

8.2.4.2 Direct2D

- 장점: 앞으로 계속 쓰일, 주목받을 기술이며, DirectX 기술과 호환성이 좋음.
- 단점: 현재 개발자원의 숙달도가 떨어짐

8.2.5 제작 스케쥴 (최초 상정)

Task Name	Duration	Start	Finish
최초 개발제안서 작성	2 days	Sat 14-09-13	Sun 14-09-14
면담 후 스케쥴 정립	1 day	Mon 14-09-15	Mon 14-09-15
프로그램 디자인	**4 days**	**Tue 14-09-16**	**Fri 14-09-19**
프로그래밍 스펙 및 필요한 기술 정리	1 day	Tue 14-09-16	Tue 14-09-16
기술적 스펙에 따른 프로토타입 펑션 개발	1 day	Tue 14-09-16	Tue 14-09-16
프로토타입 스펙 재확인	2 days	Wed 14-09-17	Thu 14-09-18
해당 디자인으로 개발 지속여부 확인/면담	2 days	Thu 14-09-18	Fri 14-09-19
디자인 완료	1 day	Fri 14-09-19	Fri 14-09-19
오브젝트 개발	**8 days**	**Fri 14-09-19**	**Tue 14-09-30**
Player(컨트롤에 따른 Update)			
Monster(AI StateMachine, update)			
월드 개발	**11 days**	**Wed 14-10-01**	**Wed 14-10-15**
KeyControl			
TileMap			
Hitbox(사용여부에 따라)			
기본 WorldMap(Tile 적용 없이)			
Camera 개발	**7 days**	**Thu 14-10-16**	**Fri 14-10-24**
Player 이동에 따라가는 카메라			
Testing Stage 및 Additional Module 개발	**8 days**	**Mon 14-10-27**	**Wed 14-11-05**
시간 가용시 추가요소(네트워크, 퀘스트, 툴 등등)			
전체 Debug	**6 days**	**Thu 14-11-06**	**Thu 14-11-13**
디버깅 기간 (위 요소에 대한 자체 QA 과정 실시)			
Qaulity Assure	**4 days**	**Fri 14-11-14**	**Wed 14-11-19**
주변 지인 동한 QA 과정 실시			

8.3 부록 3 확정 개발 계획

- 09152014 미팅결과,
- 어렸을 때 재미있게 플레이 했고, 상대적으로 그래픽 리소스가 많이 공개되어 있는 Ragnarok Online 및 Ragnarok Battle Adventure 의 리소스[9][10]를 이용한 키보드 플레이형 Action Role Playing Game.
- 8.2.2 항목의 제안 중, 마지막 항목, 8.2.2.3, 2.5D –특히 Isometric-기술을 이용한 게임을 만들기로 확정.
- 8.2.3 항목에 있어서, 최대한 가능한 모든 기술을 집약한 기술시연형 포트폴리오를 제작하기로 확정.
- 마지막 고려사항이었던 WinApi 와 D2D 선택에 관한 것은, 현재 기술의 발달을 따라가기 위해 D2D 를 사용하는 것으로 확정.
- 제작 일정은 유동적으로 조정하되, 차후 MS Project 를 통한 자원관리 실시.

[9] 라그나로크 온라인 스프라이트 : http://rosprites.blogspot.kr/

[10] 라그나로크 배틀 어드벤쳐 : http://spritedatabase.net/game/1738

8.4 개발 일지

https://github.com/arkiny/DreamCoast2D/wiki/September-Working-Diary 참조