RankPin

Unity Ranking Service

RankPin

Unity Ranking Service

Introduction

RankPin은 간단한 설정으로 유니티에서 쉽게 랭킹 시스템을 적용하기 위해 만들어졌다. 서버 개발의 부담을 없애고 적은 비용으로 Unity Application 및 Game에 쉽고 빠르게 적용할 수 있도록 제작되어 졌다. 이제 여러분은 몇 번의 클릭으로 랭킹 시스템을 쉽게 적용할 수 있다.

RankPin의 기능

자신의 정보를 서버에 저장하는 업데이트와 서버에 저장된 정보를 바탕으로 랭킹을 가져오는 기능으로 나뉘어진다.

서버에 저장하는 기능

- 유저 데이터 : 이름 및 이미지 경로 등 랭킹 UI에서 필요한 정보
- 스코어 : 랭킹에 사용 될 스코어 정보. 스코어는 누적된 Score를 업로드 한다.
- 친구리스트 : 자신과 친구로 맺어진 친구리스트의 User Id를 리스트로 만들어 업로드 한다.

서버에서 랭킹을 가져오는 기능

- 나의 랭킹 : 전체 유저 중 나의 랭킹, 친구 중 나의 랭킹
- 전체 랭킹 : 총 누적 전체 유저 랭킹, 주간 단위 전체 유저 랭킹
- 친구 랭킹 : 총 누적 친구 랭킹, 주간 단위 친구 랭킹

* 주간 랭킹은 매주 월요일에 갱신된다.

Getting Started

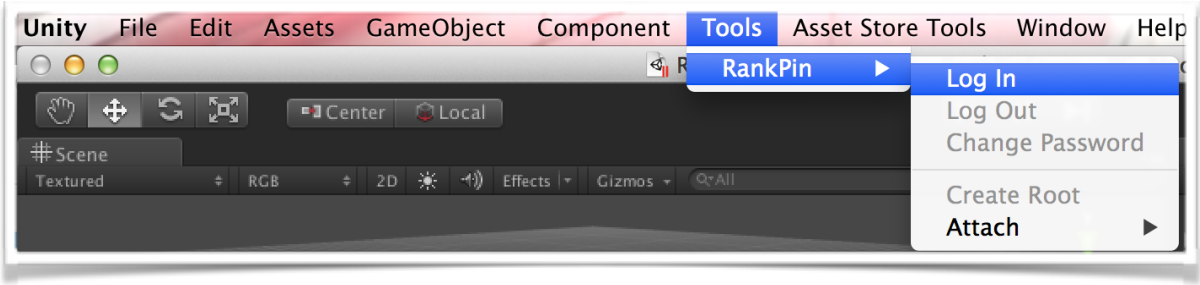
1. Log In / New Account : 랭킹 서버에 계정을 생성하거나 로그인
2. Application 생성 : 랭킹 서버에 Application을 생성
3. Root 생성 및 Application 설정 : 업데이트 및 랭킹에 사용될 Root GameObject 생성
4. Update 설정 : 유저 데이터, 스코어, 친구리스트 업데이트 설정
5. 랭킹 가져오기 : 나의 랭킹, 친구랭킹, 전체 랭킹, 주간 랭킹 등

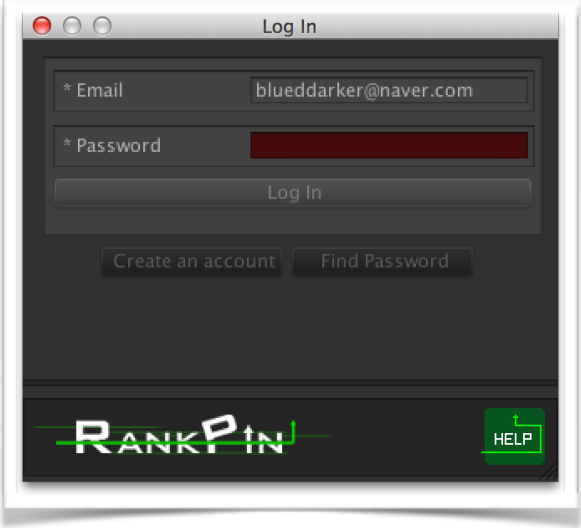
1. Log In / New Account

이메일 주소와 비밀번호를 통하여 로그인 및 계정을 생성한다. 하나의 계정을 생성하면 여러개의 Applicaiton을 생성할 수 있다.

RankPin 메뉴에서 Log In을 선택하면 다음과 같다.

로그인





* Password : 문자 + 숫자를 조합한 8글자 이상을 입력

* Find Password : 비밀 번호 찾기

(Email란에 가입당시 이메일을 입력하여야 한다.)

계정 생성

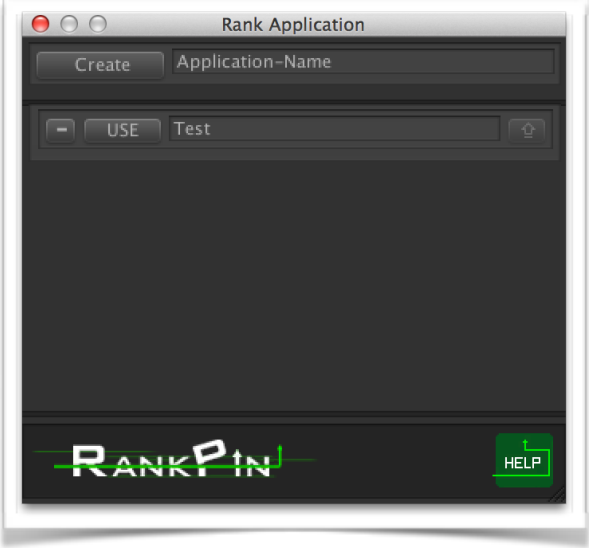
* Email : 활성화된 이메일 주소

(이메일 주소는 비밀번호를 찾는 경우 사용되므로 활성화된 주소를 사용하여야 한다)

2. Application 생성

랭킹을 적용하기 위해서는 Application을 생성해야 한다. 하나의 계정에는 여러개의 Application을 생성할 수 있다.

로그인을 수행하면 Application을 생성이 가능한 화면이 나타난다.



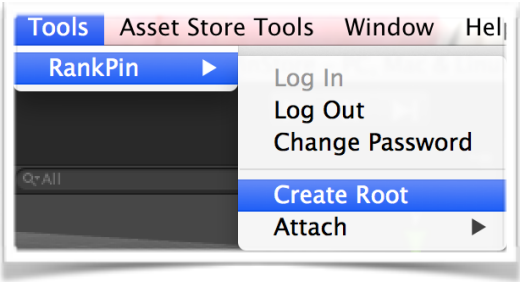
* [-] : Application 을 삭제한다. (삭제 시 모든 데이터가 서버에서 삭제된다.)

* USE : 해당 Application으로 설정한다.

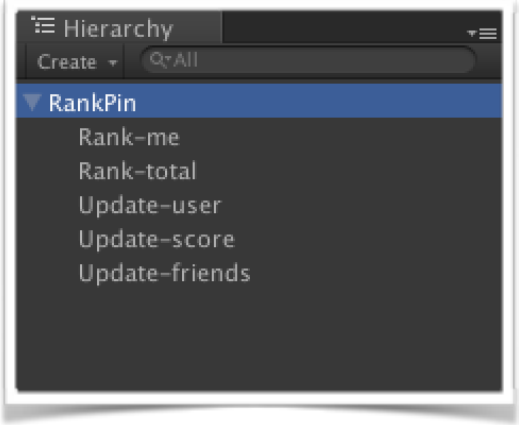
* Application 이름 변경

3. Root 생성 및 Application 설정

랭킹을 이용하기 위해서는 Rank Root를 생성해야 한다.



상단의 RankPin 메뉴에서 Create Root를 선택한다.



Hierarchy 에 다음과 같이 GameObject가 자동적으로 생성된다.

- RankApp

Application 정보를 관리하고 데이터 업데이트 및 랭킹 데이터를 주관하는 Root GameObject이다.

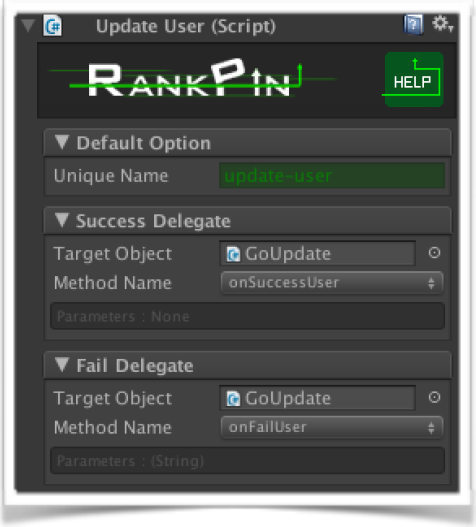
* App Name 설정 : Application을 선택

* Time out : 서버와의 통신 시 해당 시간 동안 응답이 없는 경우 에러로 처리

* User Dynamic Option : 소스코드에서 User Id를 설정 한다. (디버깅 정보)

- Update user

나의 정보를 업데이트 할 때 사용되는 GameObject이다.



* Success Delegate : 서버와 송수신이 완료 된 후 성공 정보를 받기 위해 설정

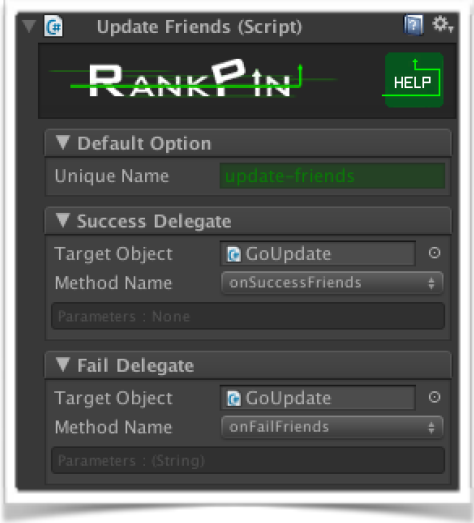
* Fail Delegate : 서버와 송수신이 완료 된 후 실패 정보를 받기 위해 설정

- Update score

나의 score 정보를 업데이트 할 때 사용되는 GameObject이다.

- Update friends

친구 리스트의 정보를 업데이트 할 때 사용되는 GameObject이다.



- Rank me

나의 랭킹 정보를 가져오는 기능을 수행하는 GameObject이다.

* User Pool

  - GLOBAL : 전체 유저 중 나의 랭킹

  - FRIENDS : 친구 중 나의 랭킹

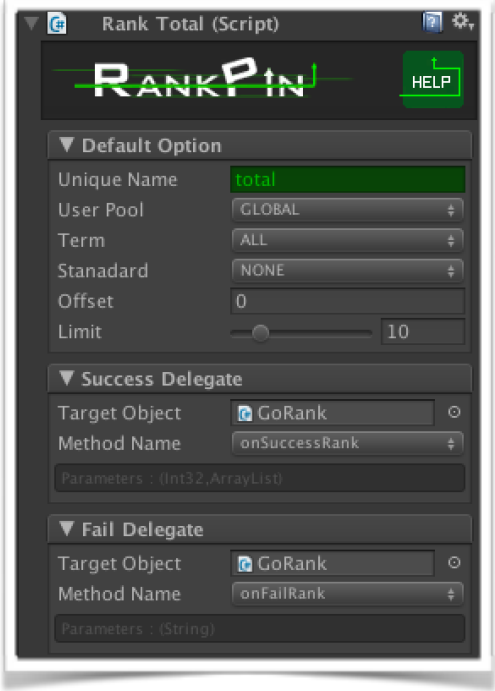
* Term

  - ALL : 전체 누적 랭킹

  - WEEK : 주간 랭킹

- Rank total

랭킹 리스트를 자져오는 기능을 수행하는 GameObject이다.



* Standard

  - NONE : Offset에서 Limit까지 랭킹 리스트를 가져온다

  - ME : 나를 기준으로 위/아래의 랭킹 리스트를 가져온다.

* Offset : 랭킹 Offset

* Limit : 가져올 랭킹 데이터의 유저 수

4. Update 설정

서버에 나의 정보를 업데이트 하는 기능을 수행한다.

- 유저 데이터 업데이트

나의 정보를 서버에 업데이트 한다. 업데이트를 할때에는 유일한 나의 아이디가 필요하다.

또한 유저 데이터는 Hashtable의 형태로 여러분이 필요한 정보를 올릴수 있다.

(내부적으로 Hashtable 데이터를 JSON 형식으로 변환되며, JSON의 길이가 500byte 이상은 서버에 저장되지 않는다.)

* Sample(RankAppUpdate.cs 참조)

Request

```
private void updateUser()
{
	Hashtable data = new Hashtable();
	data.Add("name", "Daniel");
	data.Add("url", “http://xxx.png“);
	this.updateUser(data);
}
```

Response

```
public override void onSuccessUser()
{
	base.onSuccessUser();
	Debug.Log("Update user informations.!");
}
public override void onFailUser(string message)
{
	base.onFailUser(message);
	Debug.LogWarning(message);
}
```

- 스코어 업데이트

여러분이 랭킹에 사용될 스코어 정보를 서버에 업데이트 하는 기능을 수행한다.

해당 스코어는 전체 랭킹, 주간 랭킹, 친구 랭킹에 사용되는 데이터이다.

(스코어 데이터는 누적 스코어를 서버에 올리면 된다. 서버에서 주간 랭킹을 구할 경우 자동으로 해당 주간의 스코어를 기준으로 구한다.)

* Sample(RankAppUpdate.cs 참조)

Request

```
private void updateScore()
{
	int score = 102345;
	this.updateScore((uint)score);
}
```

Response

```
public override void onSuccessScore()
{
	base.onSuccessScore();
	Debug.Log("Update score.!");
}
public override void onFailScore(string message)
{
	base.onFailScore(message);
	Debug.LogWarning("message);
}
```

- 친구 리스트 업데이트

친구 랭킹을 이용하기 위해서는 여러분의 친구 리스트를 서버에 업로드 해야 한다. 물론 친구랭킹을 이용하지 않을 경우에는 해당 정보를 서버에 업로드하지 않아도 된다.

친구 리스트는 친구 리스트의 User Id만 ArrayList로 서버에 업데이트 하면 된다.

* Sample(RankAppUpdate.cs 참조)

Request

```
private void updateFriends()
{
	ArrayList friends = new ArrayList();
	friends.Add("33");
	friends.Add("367");
	friends.Add("369");
	this.updateFriends(friends);
}
```

Response

```
public override void onSuccessFriends()
{
	base.onSuccessFriends();
	Debug.Log("Update friends list.!");
}
public override void onFailFriends(string message)
{
	base.onFailFriends(message);
	Debug.LogWarning(message);
}
```

5. 랭킹 가져오기

서버에 랭킹 정보를 가져오는 기능을 수행한다.

- 나의 랭킹

나의 랭킹 정보를 가져온다. 전체 및 주간에서의 나의 랭킹 정보를 가져올 수 있으며, 친구 중에서 나의 랭킹 정보도 가져올 수 있다.

* Sample(RankAppRank.cs 참조)

Request

```
private void requestRankMe()
{
	this.rankMe("me");
}
```

Response

```
public override void onSuccesMe(int total, int rank, int score, Hashtable data)
{
	base.onSuccesMe(total, rank, score, data);
	Debug.Log(string.Format("total:{0},rank:{1},score:{2}",total,rank,score));
}
public override void onFailMe(string message)
{
	base.onFailMe(message);
	Debug.LogWarning(message);
}
```

- 전체 랭킹

모든 유저에서 랭킹 정보를 가져온다.

랭킹 정보를 가져올 때에는 offset과 limit를 설정할 수 있다.

- offset : 랭킹 offset
- limit : 가져올 랭킹 데이터의 유저 수 (최대 50개까지 설정 가능)

* Sample(RankAppRank.cs 참조)

Request

```
private void requestRank()
{
	this.rank("total");
}
```

Response

```
public override void onSuccessRank(int total, ArrayList users)
{
	base.onSuccessRank(total, users);
	Debug.Log(string.Format("total:{0}", total));
	foreach(Hashtable user in users)
	{
		HashtableHelper.print("Rank", user);
		Hashtable data = (Hashtable)user[RankPin.RankConstants.KEY_DATA];
		if(data == null)
			continue;
		HashtableHelper.print("Data", data);
	}
}
public override void onFailRank(string message)
{
	base.onFailRank(message);
	Debug.LogWarning(message);
}
```

- 친구 랭킹

친구의 랭킹 정보를 가져온다.

* Sample(RankAppSample.cs 참조)

Request

```
private void requestRank()
{
	this.rank("friends");
}
```

Response

```
public override void onSuccessRank(int total, ArrayList users)
{
	base.onSuccessRank(total, users);
	Debug.Log(string.Format("total:{0}", total));
}
public override void onFailRank(string message)
{
	base.onFailRank(message);
	Debug.LogWarning(message);
}
```

- 주간 랭킹

주간 단위의 전체 및 친구 랭킹을 가져온다. 매주 월요일에 랭킹 데이터는 갱신된다.

* Sample(RankAppSample.cs 참조)

Request

```
private void requestRank()
{
	this.rank("week");
}
```

Response

```
public override void onSuccessRank(int total, ArrayList users)
{
      base.onSuccessRank(total, users);
      Debug.Log(string.Format("total:{0}", total));
}
public override void onFailRank(string message)
{
      base.onFailRank(message);
      Debug.LogWarning(message);
}
```

Price

RankPin은 무료로 제공 된다.

Samples

1. Simple

RankApp 를 사용하지 않고 간단하게 랭킹을 적용하는 예제이다.

- SimpleUpdate.cs : 유저데이터, 스코어, 친구리스트를 업데이트하는 예제 코드
- SimpeRank.cs : 나의 랭킹, 전체 랭킹, 주간 랭킹 등을 가져오는 예제 코드

2. RankApp_Simple

RankApp를 사용하여 랭킹을 적용하는 예제이다. Create Root를 통하여 Root를 생성한다.

- RankAppUpdate.cs : 유저데이터, 스코어, 친구리스트를 업데이트하는 예제 코드
- RankAppRank.cs : 나의 랭킹, 전체 랭킹, 주간 랭킹 등을 가져오는 예제 코드

3. RankApp_GUI

RankApp를 사용하여 GUI에 정보를 출력하는 예제이다.

- RankAppSample.cs : 업데이트 및 랭킹 정보를 가져오는 예제 코드
- RankSampleGUI.cs : 랭킹 정보를 화면에 출력하는 예제 코드