React Blockchain for Unity3D Games

ZPT [Zero Protocol Team]

INDEX

- Who we are
- Concept
    - Architects
- Advantages
- prototype
- Demo
- README
- E.O.D

Who we are?

Team 소개

ZPT

ZPT is Zero Protocol Team. We are a project team that contributes to the web3 ecosystem with a decentralized spirit and open source, without being bound by regulations.

Zero Protocol Team

ZPT는 Zero Protocol Team의 약자입니다. 우리는 규제에 혁신적이고 유연하게 문제를 해결하며 탈중앙화와 오픈 소스를 통해 웹3 생태계에 기여하는 프로젝트 팀입니다.

Concept

React base Wallet support

{Our library is designed to alleviate the complexity of wallet implementation in games developed with Unity3D and HTML5. It aims for seamless integration with existing wallets built on React, enabling users to easily connect to or expand upon a variety of wallet services.}

{우리의 라이브러리는 유니티3D 엔진과 HTML5로 개발된 게임에서 지갑 구현의 복잡성을 줄이기 위해 개발되었습니다. 이 라이브러리는 기존에 React로 만들어진 지갑과의 원활한 연동을 목표로 하며, 사용자들이 다양한 지갑 서비스를 쉽게 연결하거나 확장할 수 있도록 설계되었습니다.}

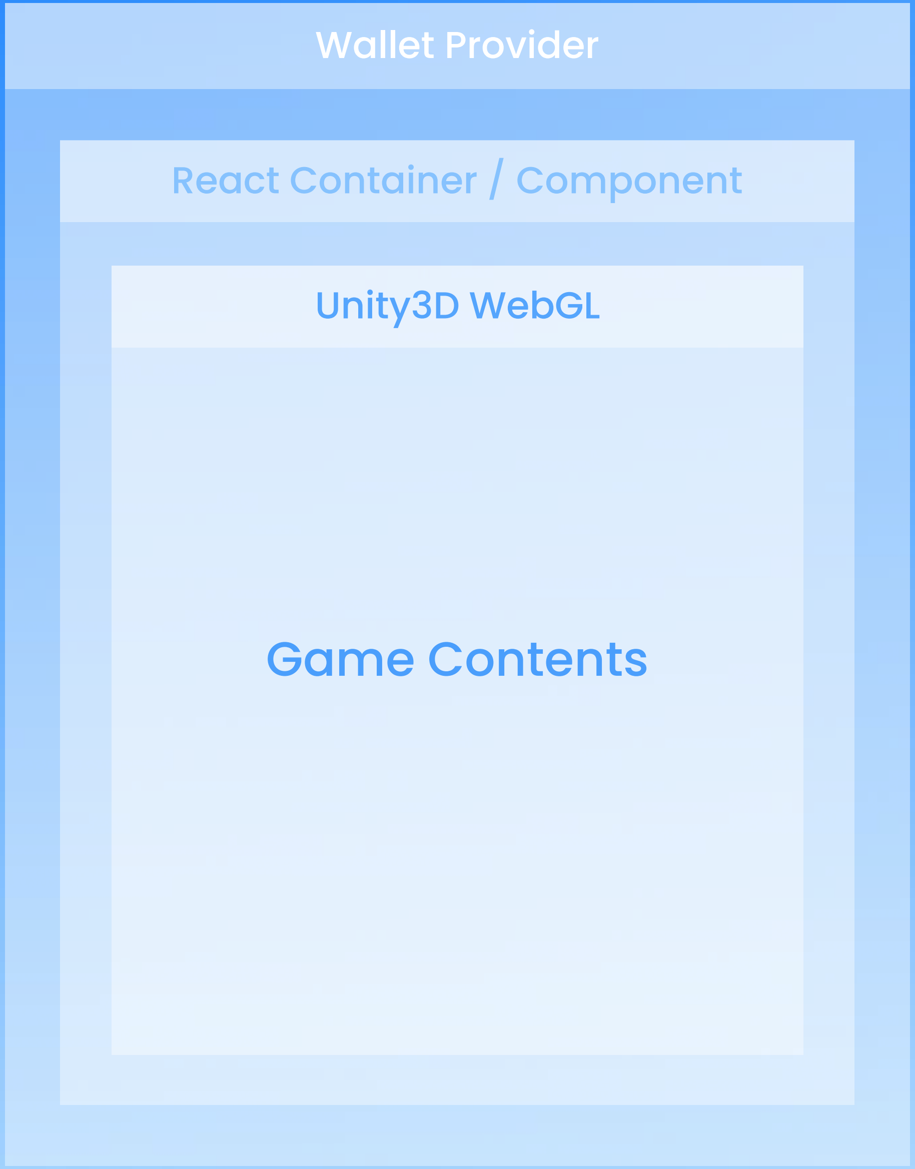
Concept

Develop Layer

Dependency libraries

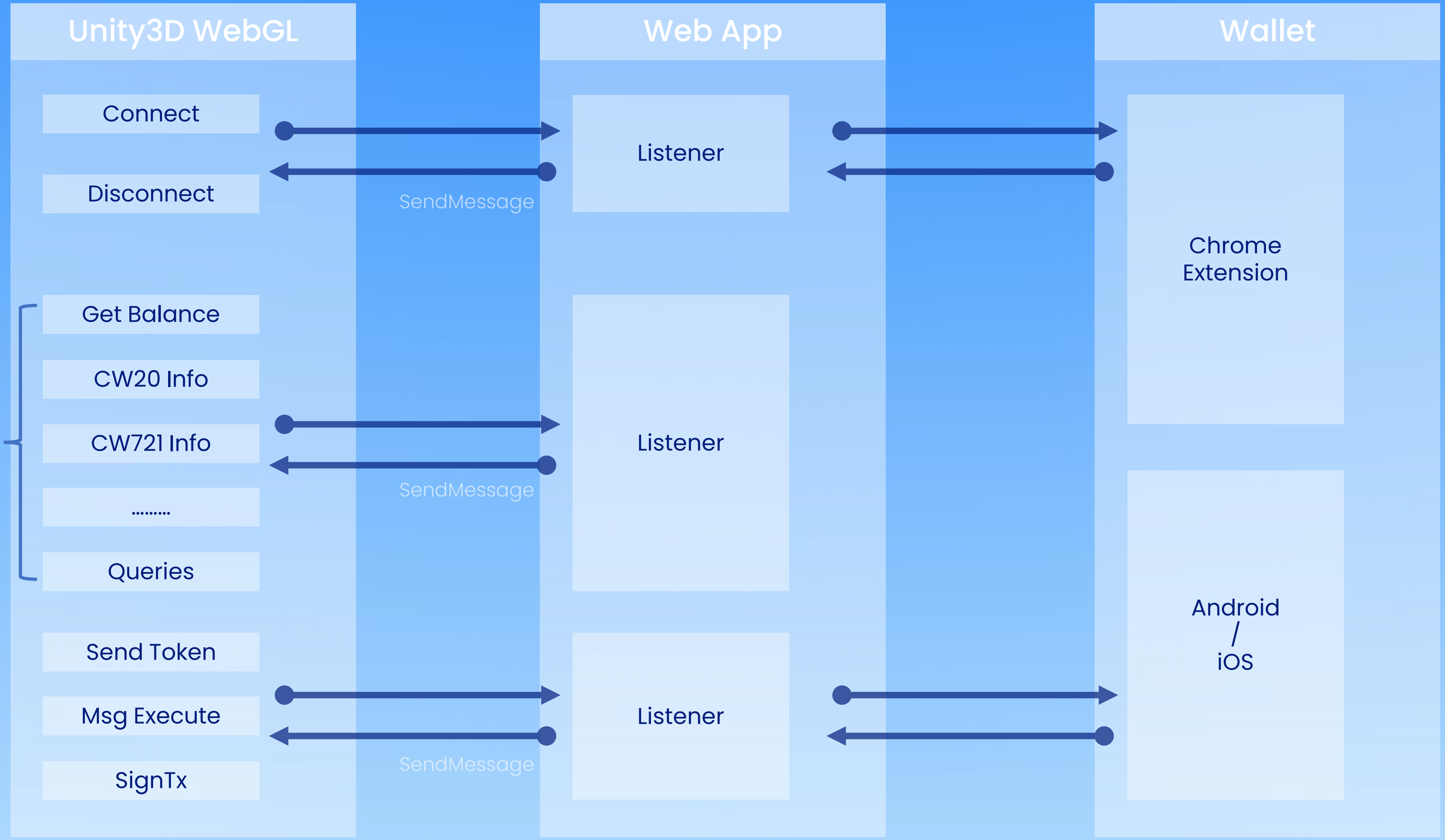
We are using the following 3 libraries for easy integration.

- @xpla/ xpla.js
- @xpla/ wallet-provider
- React-unity-webgl



Concept

Architects UML



Advantages

UML Design

Rapid Development and Deployment [신속한 개발 및 배포]
•By taking care of the complex aspects of wallet implementation, the library allows developers to create and deploy games or applications more quickly and efficiently.

Flexibility and Scalability [유연성과 확장성]
•The library makes it simple to connect with various wallet services, enhancing the project's flexibility and scalability.

Code Reusability [코드 재사용성]
•Utilizing the library means that the code related to wallet implementation can be reused in other projects. This saves development time and provides a consistent experience.

Security [보안]
•The library can rigorously handle the security aspects of wallet integration, freeing individual developers or teams from having to worry about it.

Tech Stack Compatibility [기술 스택 호환성]
•The high compatibility with existing wallets built on React means good portability across various platforms and technologies.

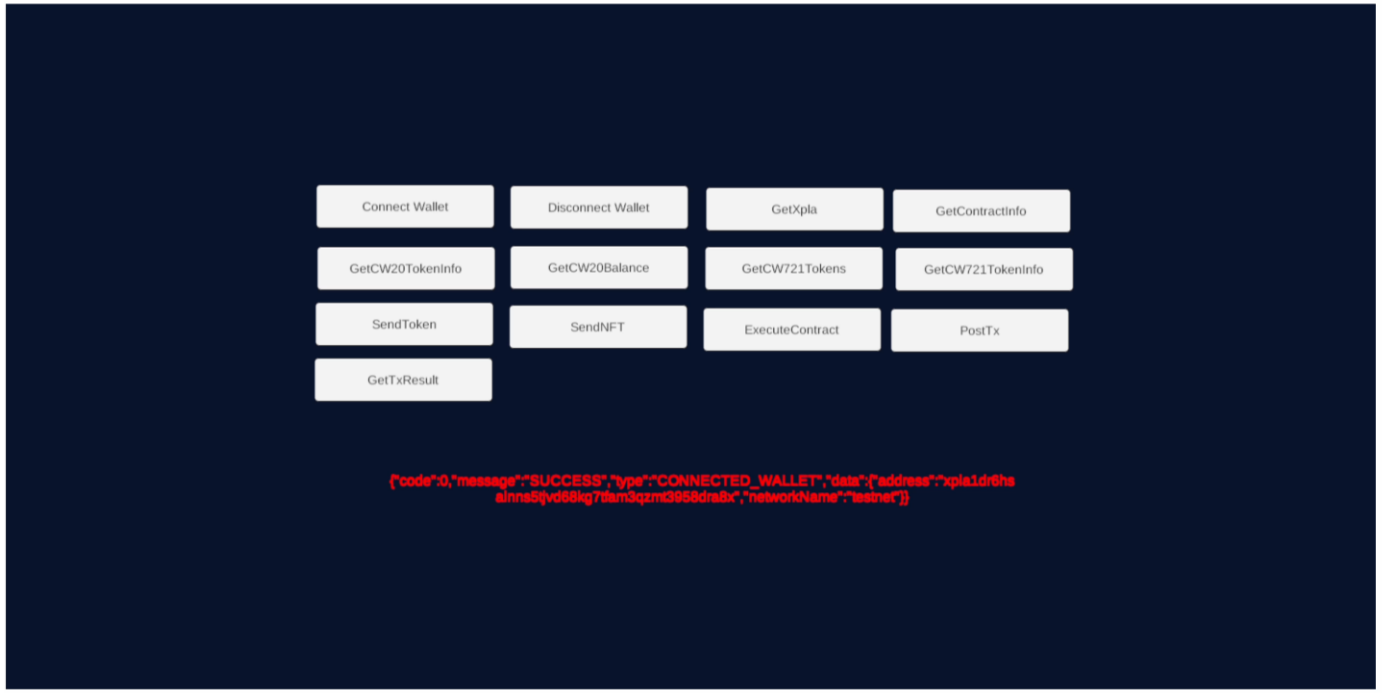
Focused Maintenance [유지보수]
•Having all the code related to wallet integration in one place makes it easier to identify and fix issues when they arise.

Improved User Experience [사용자 경험 향상]
•The ease of connecting to various wallet services improves the overall user experience.

Community and Ecosystem Support [커뮤니티 및 생태계 지원]
•If developed as open source, the library can be utilized or improved by other developers, thus having a positive impact on the broader web3 ecosystem and community.

Prototype

Demo



```
{"code":0,"message":"SUCCESS","type":"CONNECTED_WALLET","data":{"address":"xpla1dr6hs
alnns5tjvd68kg7tfam3qzmt3958dra8x","networkName":"testnet"}}
```

ZPT

Integration testing between Unity3D WebGL build and XPLA Vault is complete.

Prototype	Deploy
100%	NPM
Finish	ETA 23.09

README

~~~~markdown
## Table of Contents
- [Installation](#installation)
- [Demo](#demo)
- [How to use](#how-to-use)
- [API](#api)
- [Hook](#hook)

## Installation

```bash
npm i --save react-xpla-unity-game
```

## Demo
```bash
npm i
npx parcel sample/index.html
```

## How to use

First, to communicate from Unity3D to react, save the default WalletAPI.jslib in the
***Assets > Plugins > WebGL*** folder of the Unity3D project.

```jsx
// react
import { StrictMode } from 'react';
import { createRoot } from 'react-dom/client';
import {
XplaUnityGameWithWalletProvider,
XplaUnityGameContainer,
useRegisterEvent
} from ‘@zpt/react-xpla-unity-game';

const App = () => {
// ---------------------------------------------
// register event
// ---------------------------------------------
useRegisterEvent();

return <XplaUnityGameContainer />;
};

const config = {
loaderUrl:
'https://assets.zenaplay.com/web3games/Sample/BattleNFT40.loader.js',
dataUrl:
'https://assets.zenaplay.com/web3games/Sample/BattleNFT40.data.unityweb',
frameworkUrl:
'https://assets.zenaplay.com/web3games/Sample/BattleNFT40.framework.js.unityweb',
codeUrl:
'https://assets.zenaplay.com/web3games/Sample/BattleNFT40.wasm.unityweb',
};

const gameObjectName = 'GameManager';

const root = createRoot(document.getElementById('root') as HTMLElement);

root.render(
<>
<StrictMode>
<XplaUnityGameWithWalletProvider
config={config}
gameObjectName={gameObjectName}
>
<App />
</XplaUnityGameWithWalletProvider>
</StrictMode>
</>,
);
```
~~~~

README

Declare and use functions declared in WalletAPI.jslib in your unity3d script.

~~~
```csharp // GameManager.cs
public class GameManager : MonoBehaviour
{
[DllImport("__Internal")]
private static extern void ConnectWallet();

[DllImport("__Internal")]
private static extern void DisconnectWallet();

[DllImport("__Internal")]
private static extern void GetXplaAmount();

[DllImport("__Internal")]
private static extern void GetContractInfo(string contract);

[DllImport("__Internal")]
private static extern void GetCW20TokenInfo(string contract);

[DllImport("__Internal")]
private static extern void GetCW20Balance(string contract);

[DllImport("__Internal")]
private static extern void GetCW721Tokens(string contract);

[DllImport("__Internal")]
private static extern void GetCW721TokenInfo(string contract, string token_id);

[DllImport("__Internal")]
private static extern void PostTx();
[DllImport("__Internal")]
private static extern void GetTxResult();

[DllImport("__Internal")]
private static extern void SendToken(string token, string recipient, string amount, string
txmemo);

[DllImport("__Internal")]
private static extern void SendNFT(string contract, string recipient, string token_id,
string txmemo);

[DllImport("__Internal")]
private static extern void ExecuteContract(string contract, string execute_msg, string
txmemo);

...
...
}
```
~~~

A function to communicate from react to Unity3D must also be declared.
In this library, the function name is used as WalletEventListener.

~~~
```csharp
// GameManager.cs
public class GameManager : MonoBehaviour
{
...
...

public void WalletEventListener(string result) {
Debug.Log(result);
}

...
...
}
```

```jsx
// react
import { WALLET_EVENT_LISTENER } from '../../config';

...
...

const {
unityContextHook: { sendMessage },
gameObjectName,
} = useXplaUnityGame();

sendMessage(gameObjectName, WALLET_EVENT_LISTENER, 'test message');

...
...

```
~~~

README

~~~markdown
## API

<details>

<summary><code>&lt;XplaUnityGameWithWalletProvider&gt;</code></summary>

XplaUnityGameWithWalletProvider is composed of XPLA Vault's wallet-provider and
XplaUnityGameProvider.

XplaUnityGameWithWalletProvider includes XPLA Vault wallet by default.

The default settings are:

```jsx
// Game build files
const config = {
loaderUrl:
'https://assets.zenaplay.com/web3games/Sample/BattleNFT35.loader.js',
dataUrl: 'https://assets.zenaplay.com/web3games/Sample/BattleNFT35.data',
frameworkUrl:
'https://assets.zenaplay.com/web3games/Sample/BattleNFT35.framework.js',
codeUrl: 'https://assets.zenaplay.com/web3games/Sample/BattleNFT35.wasm',
};

// Game object name to call from React to Unity3D
const gameObjectName = 'GameManager';
```

</details>

<br />

<details>

<summary><code>&lt;XplaUnityGameProvider&gt;</code></summary>

It can be used if there is a wallet-provider of XPLA Vault that is already linked.

The default setting is the same as XplaUnityGameWithWalletProvider.

</details>

<br />

<details>

<summary><code>&lt;XplaUnityGameContainer&gt;</code></summary>

XplaUnityGameContainer is a wrapper component of react-unity-webgl's Unity component.

```jsx
// loading image or icon
const loading = <img src="IMAGE_LINK" alt="" />;

// Update Unity component style
// Changed the default width of 60% to 70%
const style: React.CSSProperties = {
width: '70%'
}

return (
<XplaUnityGameContainer loading={loading} style={style} />
)

```

</details>

## Hook
~~~

E.O.D

ZPT [ZERO PROTOCOL TEAM]
e-mail : tinywolf@gmail.com