# Unity Test Readme

1. Work content:

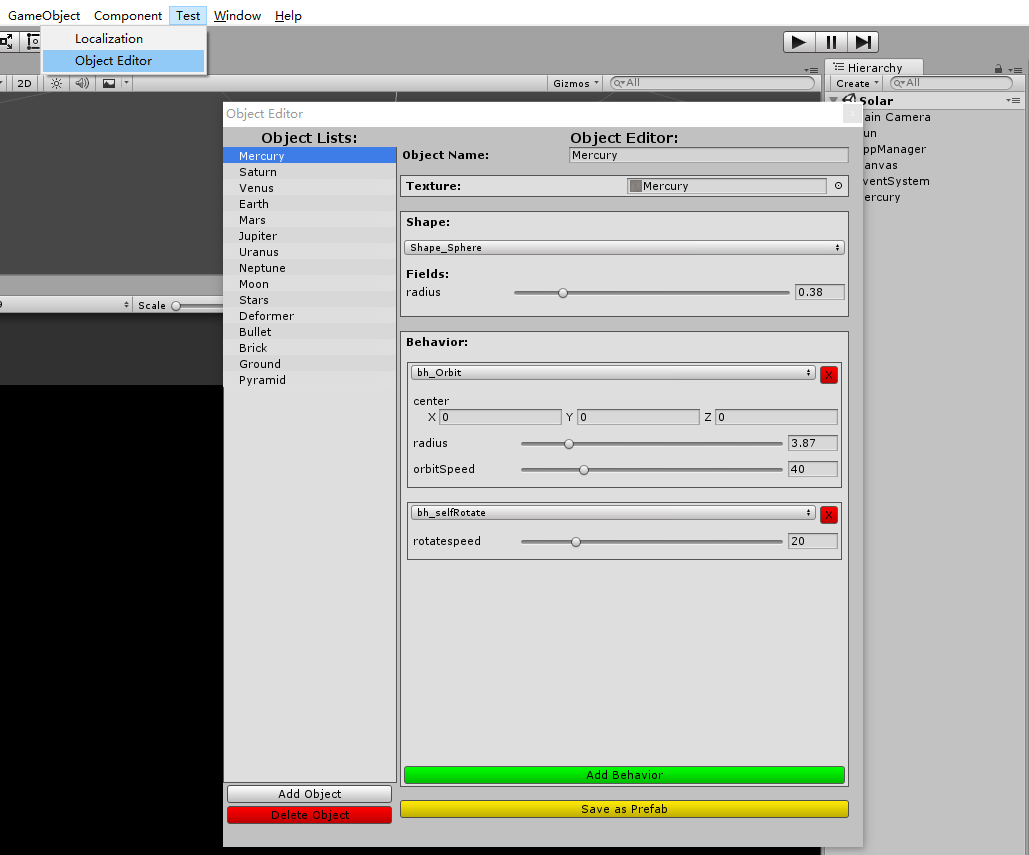
**Unity Version: 5.6.1 f1**

* 1. Tool pipeline description

In the Test, “Object Editor” Tool has been provided, for adding, deleting, editing “TestObject”.

The “TestObject” is an Entity that could been Instantiated in game, and It includes three parts: Shape Data, Texture and Behaviors.

Click Menu/Test/Object Editor to open “Object Editor” tool:



“Object Editor” consists of “Object Lists” and “Editor”.

In “Object Lists”, a “TestObject” could be adding or deleting by Button click. When a “TestObject” has been selected, “Editor” would be show on right side.

In “Editor”, a “TestObject” could be edited as follow:

1. Modify Object Name
2. Assign Texture.
3. Add, Remove, Edit “Shape” data.

Parameters of shape would be modified in runtime.

1. Add, Remove, Edit “Behavior”

Parameters of “Behavior” would be modified.

1. Save “TestObject” as a Prefab

And a GameObject which spawned by selected “TestObject” would be show in Unity Editor. If “TestObject” name, texture, shape parameters and some of “Behavior” parameters have been modified, the GameObject will be updated in runtime.

* 1. Code.

1. ObjectLists:

A ScriptableObject for storing “TestObjects” data. The asset has been saved in “Resource/ObjectLists.asset”

1. Editor/ObjectEditor:

“Object Editor” window code for editing ObjectLists.asset

1. Utilities:

Load and save property from/into string. In the test, only support 3 data type: “float”, “vector3” and Unity.Object

Instantiate “TestObject” to gameobject in game

1. Shape: Shape Data

Shape\_Sphere:

Unity Sphere PrimitiveType, using “radius” to control sphere size

Shape\_Cube:

Unity Cube PrimitiveType, using “extent” to control cube size

Shape\_Particles:

Assign particle system to generate particle in game

Shape\_Pyramid:

Simple Code Generate Mesh, using “height” and “length” to control pyramid size.

Shape\_Mesh:

Assign Mesh prefab to generate mesh Object in game.

1. bh\_xxx: TestObject behaviors

bh\_meshDeformer:

A simple mesh deform animation in runtime. Not support in editor mode yet.

bh\_Orbit:

Circular Orbit mode, using center, radius and orbitSpeed to control motion behavior. Could be executing in editor mode.

bh\_selfDestroy:

Time count down, then destroy spawned gameobject in game mode

bh\_selfRotate:

self rotation around Vector.up in local space.

bh\_setPhysics:

Add a rigidbody component to gameobject in game mode and set the rigidbody mass.

bh\_setStartDelay:

Set start delay time for particle system.

bh\_setStartTransform:

Set gameobject starting position, rotation and scale when spawning in game. Could be executing in editor mode

1. Localization: Simple Localization system

A ScriptableObject “Resources/Localization.asset” for storing languages list and updating localized text in runtime. (Show in SolarGpp Demo)

“Resources/Localization/xxx.json”: save localized content, in this test, only support English and Chinese

1. Game/SolarGpp: solar demo code
2. Game/PhysicsGpp: physics demo code
3. Game/FreeCamera: free fly camera controller in demo.
4. Demo:
5. SolarGpp:

Spawn planets one by one. The “Planet” objects are instantiated by “TestObject” which edited in “Object Editor”

Click Button “English” could switch language in demo.

Planet’s name would be showed on Screen when camera nearby.

1. PhysicsGpp:

Spawn “ground”, “deform shape”, “brick wall” one by one. All of them are instantiated by “TestObject” which edited in “Object Editor”.

Click mouse left button, A “bullet” is spawned. Try to hit the brick wall!!

1. Third Party
2. SimpleJSON:

Json parse tool for Localization system.

<https://github.com/Bunny83/SimpleJSON>

1. Editor/ListView:

UI List view, used in Object Editor.

<https://github.com/akof1314/UnityEditorListView>

1. Solar demo resource

Only used Planet texture, stars particle, sun flare

<https://github.com/sotos82/SolarSystemSimulatorGame>

1. Time Estimate

Completing the test spends 5 days.

Thinking in Test requirement and design draft: less than one day, maybe 4-5 hours

Object Editor and ObjectLists: one or two days, maybe 15 hours

Shape data and behavior: totally 2-3 hours

Demo SolarGpp: maybe less than 3 hours

Demo PhysicsGpp: maybe less than 2 hours

Localization: maybe 3 hours

Rewrite Shape data and behavior serialize: 3-4 hours

Check and fix bugs: totally 3 hours more or less

Find third party code and resource: totally 2-3 hours

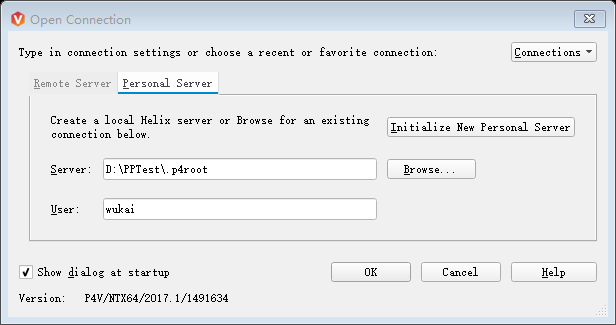
Readme.doc: 3-4 hours

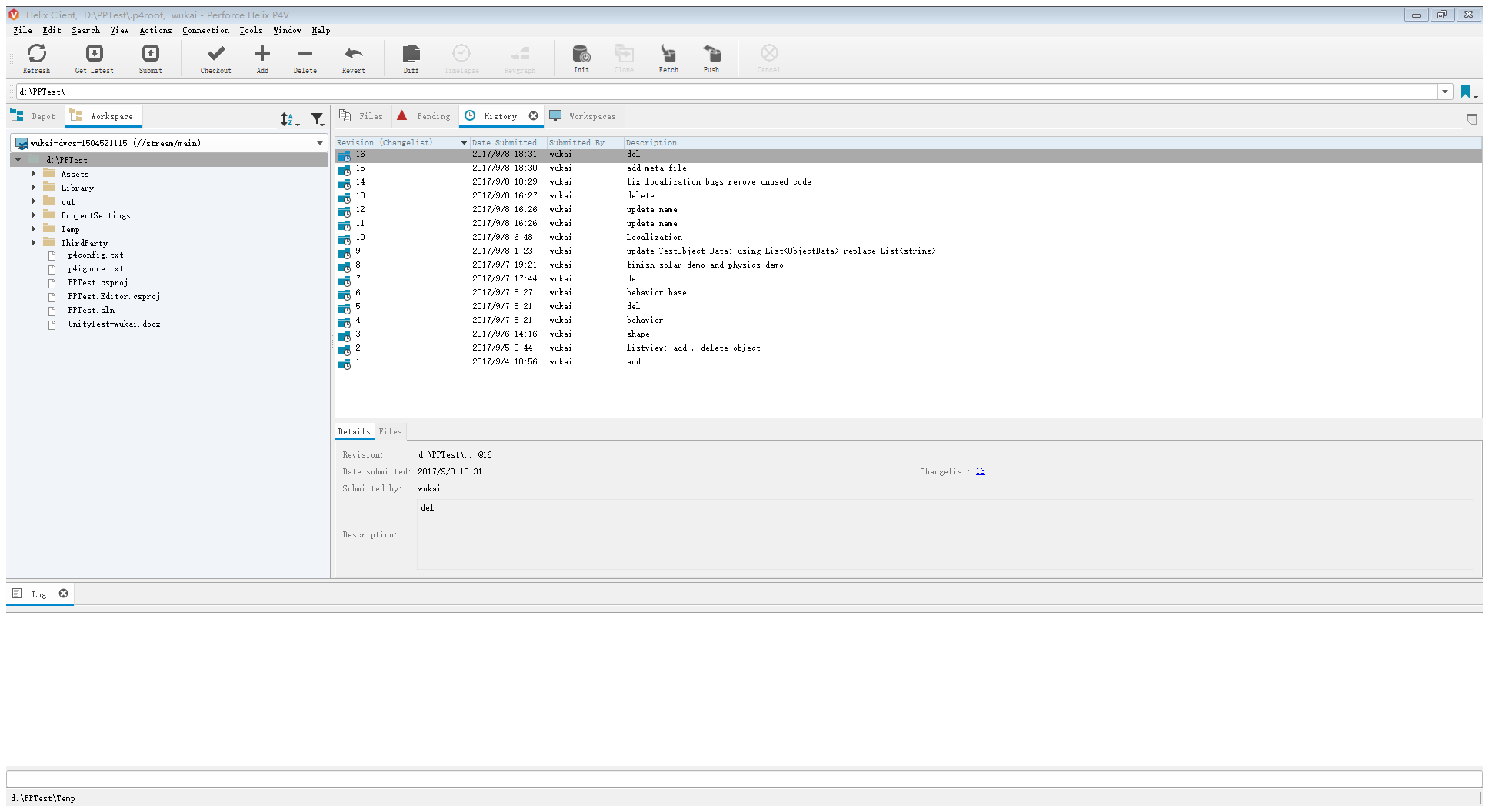
1. More:
2. Serialize:

In the test, ScriptableObject just can save simple data. If saving complex data, string stream may be a good choice. In my option, Text file could replace ScriptableObject to serialize data, such as XML or JSON

1. Perforce:

Personal Server (local depot) has been used in Test for version control on my top PC. Local depot is different with Remote depot, and there is a problem to integrate in Unity Editor. (maybe without password and port).





1. UIHUD:

In Solar Demo, planet’s name would be visible when near the camera. Simple debug GUI are used for rendering string. In the future, 3d HUD UI based on UGUI maybe better!