LZWPlib alpha 2.5

skrótowa instrukcja przygotowania projektu z użyciem silnika Unity3D

OS: Windows 7 (OpenGL), Windows 10 (DX 11/12)
Testowane na Unity: 2017.2 - 2017.3.0b10 (w starszych może nie działać)

Wstęp

Aplikacja może działać w jednym z dwóch trybów - jako serwer bądź jako klient. Serwer zajmuje się obsługą danych wejściowych (system śledzenia, joystick, klawiatura etc.) i aktualizacją stanu aplikacji, który następnie synchronizuje ze wszystkimi klientami. Klient w większości przypadków jedynie aktualizuje niezbędne elementy stanu aplikacji i zajmuje się wyrenderowaniem obrazu do wyświetlenia przez projektor/monitor.
Serwer również generuje obraz. W przypadku miniCAVE'a i MidiCAVE'a jest on wyświetlany na frontowym ekranie, zaś w przypadku BigCAVE'a - na audytorium, jeśli jest taka potrzeba.

Sprawdzenie trybu działania aplikacji:

```
using LZWPlib;
…
if (LzwpManager.Instance.isServer) { … }
```

Ogólne przygotowanie projektu

1. Zaimportować paczkę LZWPlib_a2.5_MOCK.unitypackage
2. Dodać do sceny prefab LZWPlib_a2 - musi być na samym wierzchu w hierarchii
3. W komponencie 'Lzwp Manager' nowo dodanego obiektu kliknąć przycisk 'Prepare project for CAVE'
4. Dodać do sceny jeden z prefabów - FlyController lub FPSController (lub wersję _2cam) - lub samodzielnie dodać obiekt reprezentujący jaskinię i jego zawartość, analogiczną do wymienionych prefabów
5. Synchronizacja (Lagacy Networking - https://docs.unity3d.com/Manual/NetworkReferenceGuide.html): dodać komponent NetworkView do obiektów, których pozycja i rotacja mają być synchronizowane; oprogramować synchronizację pozostałych elementów (np. poprzez mechanizm RPC)
6. Oprogramować sterowanie za pomocą Flysticków
7. Zalecane: dodać do kamer post-processing (i włączyć w nim Antialiasing) - https://www.assetstore.unity3d.com/en/#!/content/83912

Przycisk ‘Prepare project for CAVE’

- włącza obsługę stereoskopii i wsparcie dla VR
- dodaje API graficzne (DX 12, DX 11, Vulkan, OpenGL Core)
- ustawia: Run in background = true, Visible in background = true, Display resolution dialog = hidden by default

Wyświetlanie obrazu

Aby prawidłowo wygenerować obraz stereoskopowy, dopasowany do ekranów jaskini, na scenie muszą znajdować się:

a) obiekt reprezentujący umiejscowienie jaskini w świecie gry; gdy przemieszczamy się z pomocą Flysticka - przesuwamy ten obiekt
b) obiekt okularów - musi należeć do obiektu a) w hierarchii obiektów sceny; gdy przemieszczamy się fizycznie w obrębie jaskini - przesuwamy ten obiekt
c) obiekt oczu - kontener zawierający kamery (domyślnie jedną); w hierarchii należy do obiektu okularów; ‘przełączając’ osobę, dla której perspektywy dopasowywany jest obraz, obiekt oczu przenosi się do innego obiektu okularów
d) oczy - kamery, domyślnie jedna, należy do obiektu okularów

Pozycja obiektów b, c i d nie powinna być ustawiana przez użytkownika. W celu przemieszczenia kamery należy dopasować pozycję i rotację obiektu a.

Pobranie parametrów ekranu na przykładzie jego środka:

```
Vector3 pos = LZWPlib.LzwpManager.Instance.screen.centre;
```

Pole ‘Workarounds for stereo’ skryptu LzwpCamera

Podczas testów aplikacji w trybie stereoskopowym występowały problemy z renderowaniem cieni. Aby je obejść, przy starcie aplikacji modyfikowane są ustawienia kamery:

- wyłączone zostają HDR i MSAA,
- Rendering Path ustawione zostaje na DeferredShading (jeśli pojedynczy obiekt kamery renderuje obraz dla obu oczu),
- FOV na 179 stopni.

Synchronizacja

Do obsługi synchronizacji wykorzystywane są stare mechanizmy sieciowe silnika Unity - https://docs.unity3d.com/Manual/NetworkReferenceGuide.html
Do synchronizacji wykorzystywane są komponenty NetworkView. Dodanie takiego komponentu do obiektu domyślnie powoduje synchronizację jego położenia i rotacji.
Inne elementy synchronizować można poprzez zdalne wywołania procedur - RPC - https://docs.unity3d.com/Manual/net-RPCDetails.html

Śledzenie - okulary i flysticki

Aplikacja działająca jako serwer pobiera od systemu śledzenia dane dotyczące położenia i rotacji okularów oraz flystików. Dla flystików pobierane są również stany przycisków i joysticka.
Aby zsynchronizować położenie w obrębie obiektu reprezentującego jaskinię obiekt okularów lub flysticka należy dodać do takiego obiektu odpowiednio komponent GlassesTracker lub FlystickTracker (i ustawić odpowiedni indeks obiektu, liczony od 0), oraz komponent NetworkView.

Pobranie liczby dostępnych flystików i okularów:

```
int flysticksCount = LZWPlib.LzwpTracking.Instance.flysticksCount;
int glassesCount = LZWPlib.LzwpTracking.Instance.glassesCount;
```

Pobranie pozycji i rotacji:

```
glasses.transform.localPosition =
LZWPlib.LzwpTracking.Instance.glasses[0].position;
flystick.transform.localRotation =
LZWPlib.LzwpTracking.Instance.flysticks[1].rotation;
```

Sterowanie

Odczyt stanu przycisków i joysticka dla danego Flysticka (zwykle dla tego z indeksem 0):

```
using LZWPlib;

if (LzwpTracking.Instance.flysticks[flIdx].active) {...}  // czy
flystick znajduje się w zasięgu kamer śledzących?


if (LzwpTracking.Instance.flysticks[flIdx].button1.isActive) {...}
// przycisk jest wciśnięty
if (LzwpTracking.Instance.flysticks[flIdx].button1.wasPressed)
{...}  // przycisk został teraz wciśnięty (pojedynczy update)
if (LzwpTracking.Instance.flysticks[flIdx].button1.wasReleased)
{...}  // przycisk został teraz zwolniony (pojedynczy update)

// dostępne przyciski: button1, button2, button3, button4, fire,
joystick

float h =
LzwpTracking.Instance.flysticks[flIdx].joystickHorizontal; // od
-1f (lewo) do 1f (prawo)
float v = LzwpTracking.Instance.flysticks[flIdx].joystickVertical;
// od -1f (dół) do 1f (góra)
```

Symulowanie Flysticków za pomocą klawiatury

W celach testowych; można włączyć/wyłączyć w komponencie LzwpTracking.

	Flystick 0	Flystick 1
fire button	Space	Keypad0
button 1	Alpha1	Keypad1
button 2	Alpha2	Keypad2
button 3	Alpha3	Keypad3
button 4	Alpha4	Keypad4
joystick button	Alpha5	Keypad5
joystick vertical	W / S	---
joystick horizontal	A / D	---

Znane problemy

- Aplikacja wysypuje się, jeśli uruchomi się ją z włączoną stereoskopią, a na scenie znajduje się obiekt Terrain o dużej powierzchni (> 1 km^2) - w trybie z jedną kamerą, renderującą dla obojga oczu.
- Użycie Reflection Probe powoduje zawieszenie się renderowania dla jednego z oczu - w trybie z jedną kamerą, renderującą dla obojga oczu.
- Użycie jako drzewo innego niż SpeedTree obiektu w komponencie Terrain powoduje zawieszenie się renderowania dla jednego z oczu podczas rotacji kamery - w trybie z dwiema kamerami renderującymi dla poszczególnych oczu.

Uruchomienie aplikacji

- na własnym komputerze - można uruchomić w edytorze lub jako samodzielną aplikację
  - aby przetestować synchronizację można uruchomić aplikację dwa razy: jako samodzielny build oraz w edytorze, jedną z nich jako serwer, drugą jako klient (Default role - Editor / Standalone w komponencie LzwpManager)
  - dla prostego testu stereoskopii można wybrać tryb podzielonego ekranu (Mode - split) w komponencie LzwpStereoscopy
- jaskinia - przekopiować folder z aplikacją do katalogu aplikacji (skrót znajduje się na pulpicie - `v1.1`), podmienić pliki dll (LZWPlib_a2.dll i UnityTrackingPlugin.dll), uruchomić aplikację z poziomu startera - `Cave Visualizations Starter v1.1`

Logi

Nowe wersje Unity zapisują logi do katalogu:
`C:\Users\<username>\AppData\LocalLow\<CompanyName>\<ProductName>\`
czyli przykładowo:
`C:\Users\user\AppData\LocalLow\DefaultCompany\MyTestApplication\output_log.txt`
Skrót na pulpicie: `Unity 2017 logi`

Po uruchomieniu aplikacji na CAVE'ie

- aplikacja działa pod OpenGLem na systemie Windows 7, po przejściu na Windows 10 powinna działać pod DX 11
- jeśli pozostawiono zaznaczenie pola 'Workarounds for stereo' - w kamerze renderującej obraz do wyświetlenia na ekranie wyłączone zostają HDR i MSAA, Rendering Path ustawione zostaje na DeferredShading (jeśli pojedynczy obiekt kamery renderuje obraz dla obu oczu), a FOV na 179 stopni (obejście błędu związanego z cieniami)