Introduction

My project has a basis of being a platformer within a three-dimensional perspective, this is captivated via the use of multiples camera perspectives; having varied rotational offsets. The theme and narrative of my game targets an abstractive simulation of a memory-byte being lost within cyberspace (hence 'Byte'), your end objective is to complete all the levels to progress to the end scene; the byte's end objective is to return to cyberspace, but a boss-like entity must be destructed to reach this narrative end. Progression is established via the use of 'checkpoints' during respawn conditions, meanwhile each level design corresponds to a device, imagined for the byte to be 'transmitted to and from' once each end level goal is reached. In order, relative to the level design progression: a hub, a television and a mobile phone. Notably each level is unique, whereby incorporates separate colour schemes and game mechanics, all of which attempt to provide context to the backgrounding level design. An exemplification of this, is where the 'glitch blocks' within level two (television) animate by colour and transform, to correlate to the television colour bar effect relaying on of the backgrounding television object (imported from Maya, .mb). With 'glitching' one would expect randomisation, which is the concept I have pursued and executed, one example being the application of physic materials to my glitch blocks; providing bounciness to the player object relative to the point of their contact. This concept is concurrent throughout every level mechanic. I am particularly intrigued by my use of particle systems to not only reinforce the cyberspace aesthetic of the game, but to apply force and sometimes collision effects to the player object and world space environment. The variation of particle systems implemented, aims to provide mobile players with unexpectaion and to further create real-time suspense.

Resources

Regarding resources, I had used the series of Unity, Autodesk Maya, Adobe Photoshop, Adobe After Effects and Audacity applications to create my game and its assets entwined; to note all assets and scripts within the game other than audio and video-based material maps were all originally created by myself. To establish the 'cyberspace' theme, I had exported my 'cube' model, rig and animation set from Autodesk Maya; this formed a part of my 3D Modelling module submission, and hence my reason for choice. The noticeable difference between the export is that there is no mesh extrapolation which was present in Maya. Adobe Photoshop was utilised for the creation of all my sprites including, user interface buttons, particle system texture sprites and text-based sprites used for titling purposes (start and end scenes), this was necessary for me to render and reinforce the 'glitch' aesthetic. Meanwhile, Adobe After Effects suited my needs for video file conversion from AVI to .MOV(H.264 codec support), this was purposed to be supported by Unity's video codec range; and moreover enabled the video clip to be reduced in length and also file size. Specifically, its use was allocated for the 'matrix' effect, which has been applicated on glitch blocks and the backgrounding mobile phone object, within game level three (plays on awake). Also, Audacity had use during audio clipping, whereby the user interface buttons use sounds when 'on-click' conditions are met; I required the audio to play immediately, clipping provided this responsiveness. Also, Audacity had further application in the creation of audio files with seamless looping, all of which has enhanced immersion; clipping was used once more, as well as fade-in and fade-in effects to achieve such. Lastly, Unity was made use of as my directed game engine, in which has enabled me to utilise my game assets and scripts conjunctively, to compile a fully functioning 'cyberspace-platformer' themed 'mobile-deliverable' game.

Process

Relating to process, I had originally imported the cube model from my 3D Modelling module submission Maya project; this formed the basis of my playable character, as previously mentioned, the rig and animation set all included from the export. Within Unity I had initially created all my folders assigned for my project assets, materials, models and prefabs etc; progressively, I approached creating my levels (scene) in succession of completing the previous one. In which I had systematically created the levels static platforms and barrier system firstly, then incorporated projectiles, mobile platforms, particle systems, lighting, models, cameras and sounds following; this approach sustained clarity on each level design intention. Such approach had enabled me to adaptively create and included pre-existing game mechanics, as well as prefabs, having enhanced the production speed of my game project. Furthermore, this progressive approach had allowed me to create, adjust and implement scripts more efficiently; more functions and variables were included overtime, to maximise script potential, I had also adopted the use of 'tags' to enhance ease of functionality, relative to collisions and triggers. The ability to create prefab instances of game objects allowed my game to be symmetrically 'mobile deliverable', whereby my user interface as an example only required to be made once and ensured an equal aspect ratio and screen positioning for all the mobile 'buttons', this was further aided with the use of anchoring. Anchoring the button positions ensures the button position and scaling remains proportional for any screen resolution, which was one mobile consideration I had wanted to accomplish. Once basic user interface functionality was achieved, I had created my button sprites within Adobe Photoshop to comply to my game theme aesthetically, whilst acknowledging that it was unnecessary. For additional mobile consideration, manging my assets was essential for memory and graphical performance. Hence the use of Adobe After Effects aimed to reduce memory and frame rate compromise, via reducing the file sizes for video-mapped materials; likewise, Audacity reduced memory usage from audio clipping. More so, Audacity has enabled my game audio to become seamless between looping and the transition between other sound effects. I believe the overall dynamic of the game has been predominantly impacted by the audio (specifically the sound cinematic), and such immerses 'mobile players' (headphones recommended). For post-production processes, I had then proceeded to test my game via creating a buildable .apk file, executable by android handsets; bugs and errors have been progressively amended, to its concurrent playable state. To note, multiple devices have been used to demonstrate working order and range of compatibility.

Reflection

In view of my game project 'Byte', I believe from a 'mobile deliverable' perspective suffices for player immersion and achieves sense of progression as a platform-based game should. This is achieved with the variation of level dynamic, abstract narrative and further compliance to the use of obstructions, checkpoint-systems and alternating perspectives relative to world space directionally (camera offsets). With acknowledgement to compatibility, I have peer-proved the games functionality across numerous android devices, and performance regarding memory and graphical output; which demonstrates playability on differing devices with differing screen resolutions, the user interface scales correctly also. In focus of what could have been done differently, I would consider not using generic platformer blocks and moving platforms to provide more originality to my game and consider additional levels before the end climax. Alternatively, I would consider texturing more game objects to fulfil a sense of world space completion. Meanwhile, more focus could have been applied to the inclusion of additional sound effects, which could have further enhanced the game dynamic; I believe the game would seem more professional-like and provide a better scope into immersion. Moreover, the use of respawn animations or text-to-display could improve game awareness, for players who become more disorientated; additions to the user interface could suffice this. Overall, I believe 'Byte' is successful, through its exploration and integration of numerous instances of camera and lighting manipulation, animation, projectiles, prefab and material creation, adherence to asset workflow, scripts, video mesh mapping, collision and interaction, touch and key inputs, the loading of levels, level design and destruction of the boss-like entity. In which case, 'Byte' has been produced as I had intended.

Appendix

Relating to aspirations for my project, I did not aspire to any concurrent games or projects externally, but rather my own 3D Modelling module project. I had wanted to integrate some element of the project into the game, as previously intrigued by the articulation of the cube's animation and visual resemblance to a generic 'computational byte'. That is how my narrative was founded and developed upon; and such the games mechanics, moveable entities, static entities and level designs revolved around. However, the level transition as a 'transmission through cyberspace' was inspired by blockbuster movie "Ralph Breaks the Internet (2018)", from which explores networking and game intervention. 'Byte' is a construct of this idea.

References

Model [assets]

- Mobile phone game object

https://www.turbosquid.com/FullPreview/Index.cfm/ID/1228851

- Television game object

https://www.turbosquid.com/FullPreview/Index.cfm/ID/1197865

Audio [assets]

- End/ start scene soundtrack

https://www.youtube.com/watch?v=fhgtJjs8pFk&t=29s

- Level soundtrack (all levels)

https://www.youtube.com/watch?v=NZ4X2Q5R7_Y

- Sound effects (all sound effects)

https://freesound.org/

Video [assets]

- Matrix video (used as a material for glitch blocks)

https://www.youtube.com/watch?v=u2oa5smqWHE

- Television colour bar effect video (used as a material for television model)

https://www.youtube.com/watch?v=DDVzCzgaQOQ

Project files

The project game files including the executable files can be accessed and downloaded from Google Drive.

- Game Files (not compressed)

https://drive.google.com/open?id=1udlynmnKouVZiECZrUnNupEm66Me5Dpi

- Game files (compressed, .zip)

https://drive.google.com/open?id=1RGdYkYirQHzf1c8iF6gitoMrwFlJGI8s

Alternate link to access and download the game files, using OneDrive.

- Game files (compressed, .zip)

https://1drv.ms/u/s!AgmezDTAnTY-iaZrv28uWkHjDAOLjA

Process with pictures

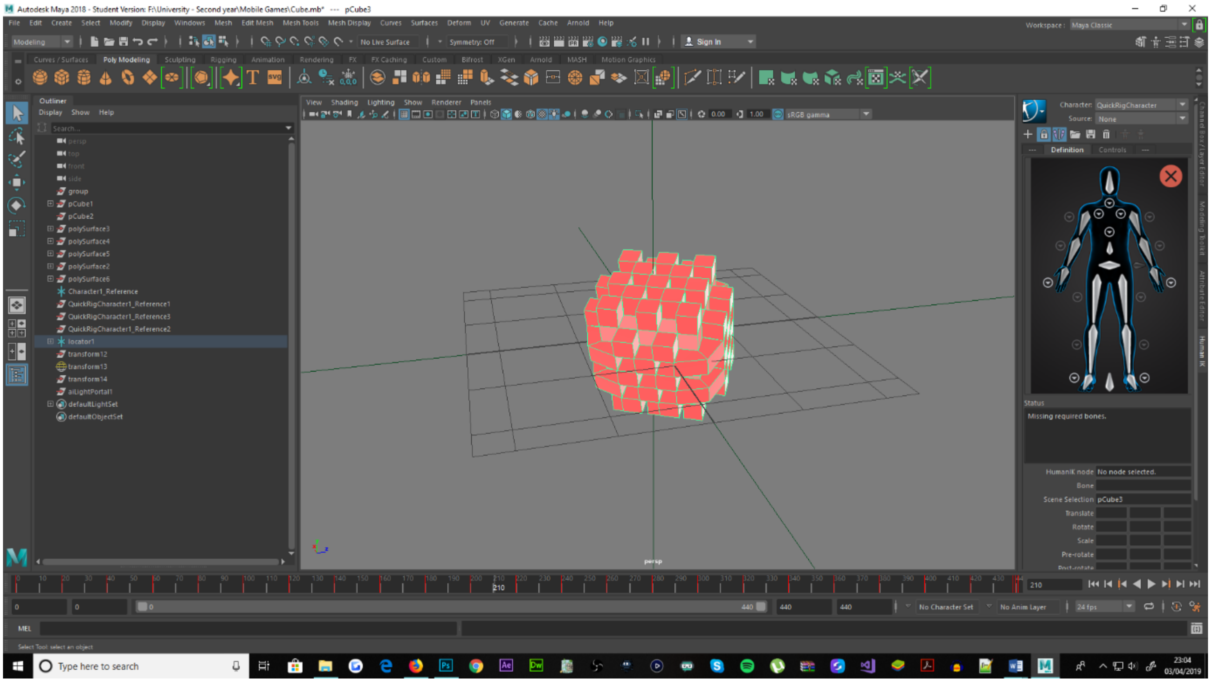


Figure 1: Maya export, cube object (3D Modelling module)

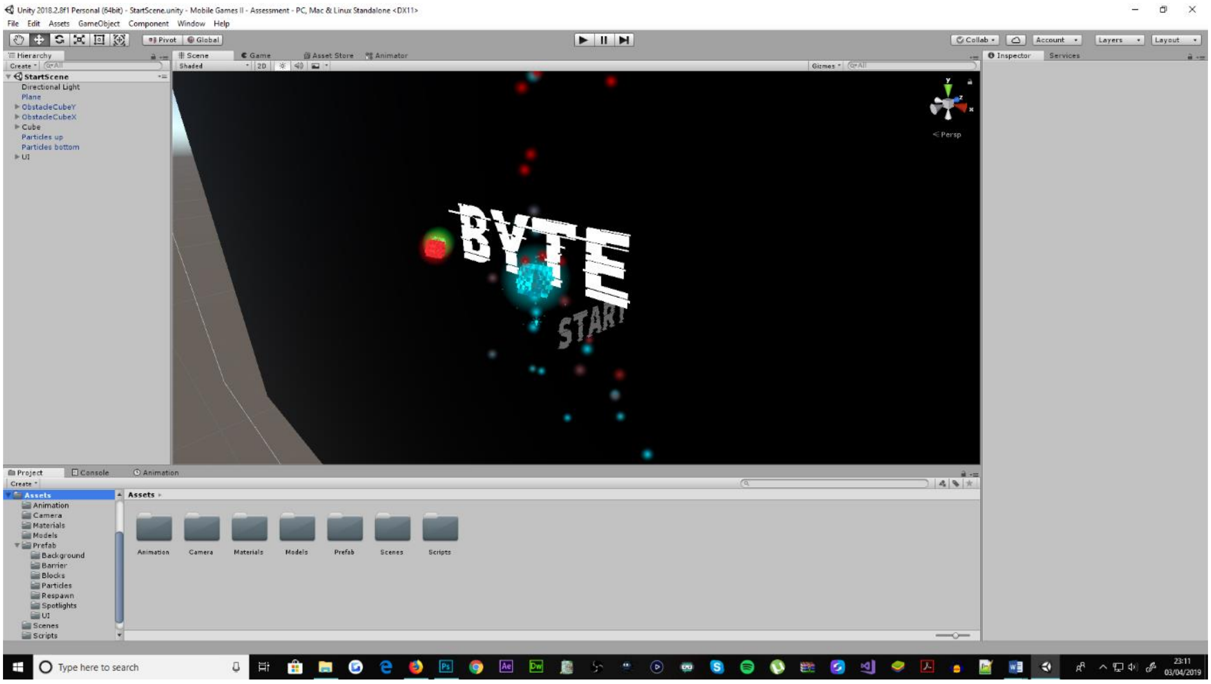
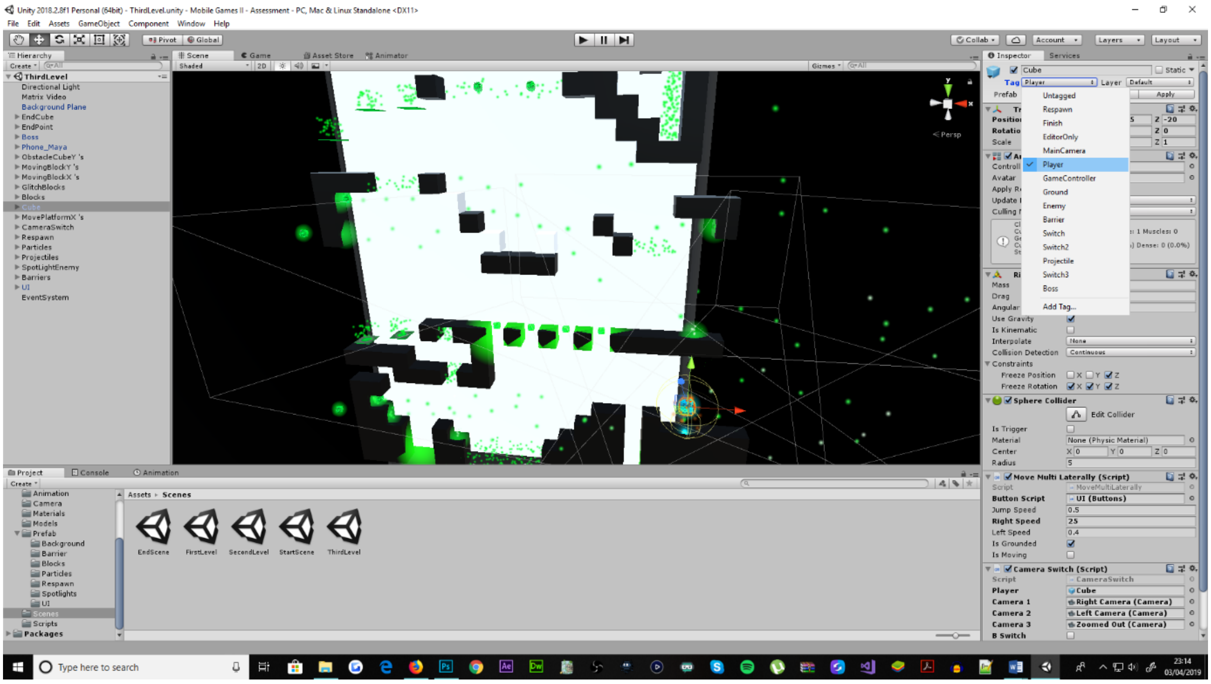


Figure 2: Folder organisation, managing asset workflow



Inspector

Cube

Tag: Player

Layer: Default

Use Gravity ☑

Is Kinematic □

Interpolate: None

Collision Detection: Continuous

Constraints

Freeze Position □ X □ Y ☑ Z

Freeze Rotation ☑ X ☑ Y ☑ Z

☑ Sphere Collider

Is Trigger □

Material: None (Physic Material)

Center: X 0 Y 0 Z 0

Radius: 5

☑ Move Multi Laterally (Script)

Script: MoveMultiLaterally

Button Script: UI (Buttons)

Jump Speed: 0.5

Right Speed: 25

Left Speed: 0.4

Is Grounded ☑

Is Moving □

☑ Camera Switch (Script)

Script: CameraSwitch

Player: Cube

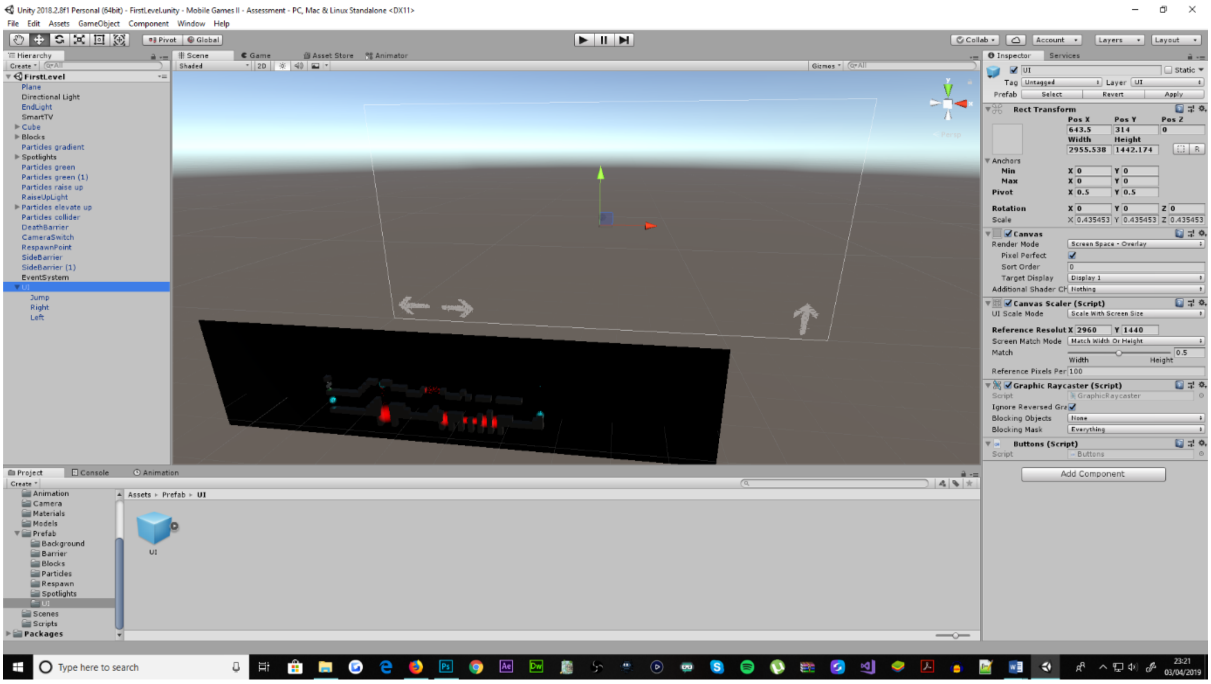
Camera 1: Right Camera (Camera)

Camera 2: Left Camera (Camera)

Camera 3: Zoomed Out (Camera)

B Switch □

Figure 3: Applying tag to game object



Inspector

UI

Tag: Untagged

Layer: UI

Rect Transform

	Pos X	Pos Y	Pos Z
	643.5	314	0
	Width	Height	
	2955.538	1442.174	
Anchors Min	X 0	Y 0	
Max	X 0	Y 0	
Pivot	X 0.5	Y 0.5	
Rotation	X 0	Y 0	Z 0
Scale	X 0.435453	Y 0.435453	Z 0.435453

☑ Canvas

Render Mode: Screen Space - Overlay

Pixel Perfect ☑

Sort Order: 0

Target Display: Display 1

Additional Shader Ch: Nothing

☑ Canvas Scaler (Script)

UI Scale Mode: Scale With Screen Size

Reference Resolut: X 2960 Y 1440

Screen Match Mode: Match Width Or Height

Match: 0.5

Reference Pixels Per: 100

☑ Graphic Raycaster (Script)

Script: GraphicRaycaster

Ignore Reversed Gra ☑

Blocking Objects: None

Blocking Mask: Everything

Buttons (Script)

Script: Buttons

Figure 4: User interface prefab

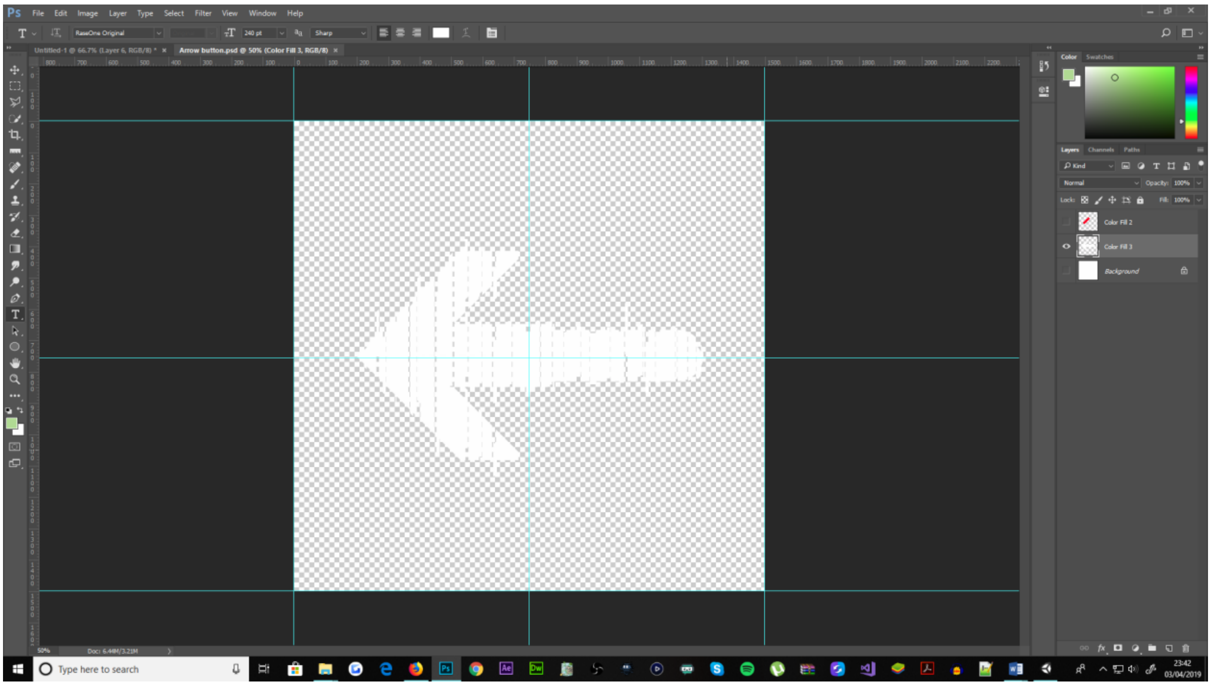

File Edit Image Layer Type Select Filter View Window Help
Arrow button.psd @ 50% (Color Fill 3, RGB/8)
Layers Channels Paths
Color Fill 2
Color Fill 3
Background
Type here to search

Figure 5: Photoshop, creation of button sprites

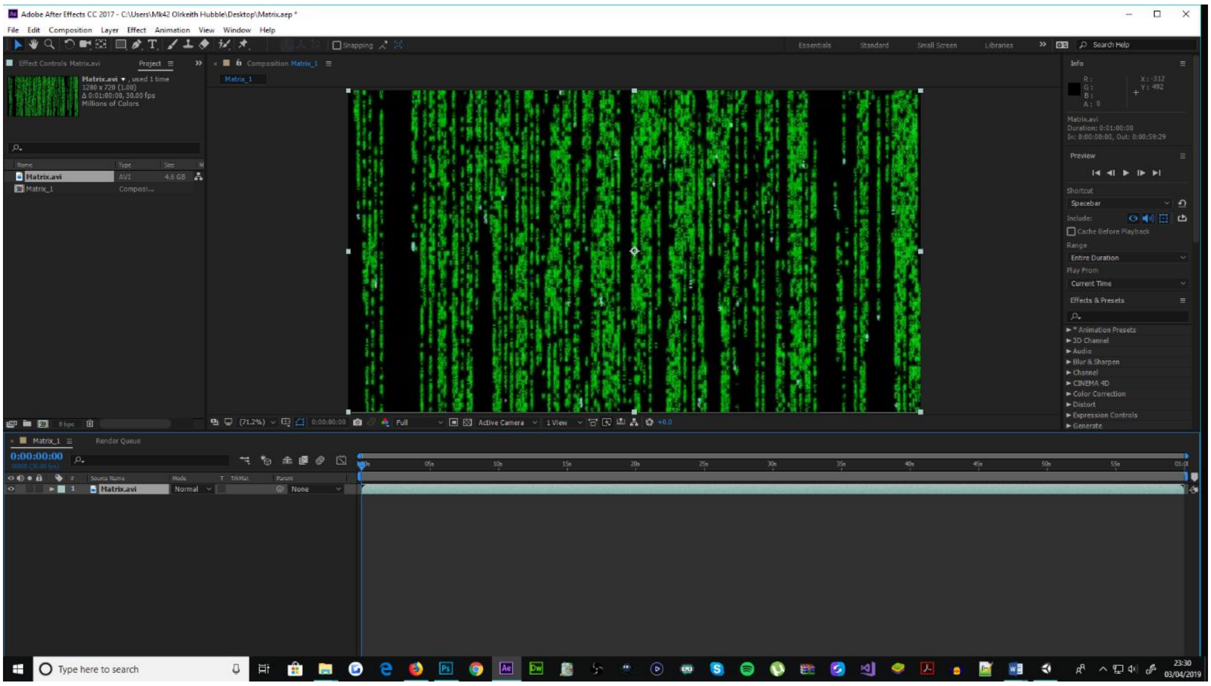
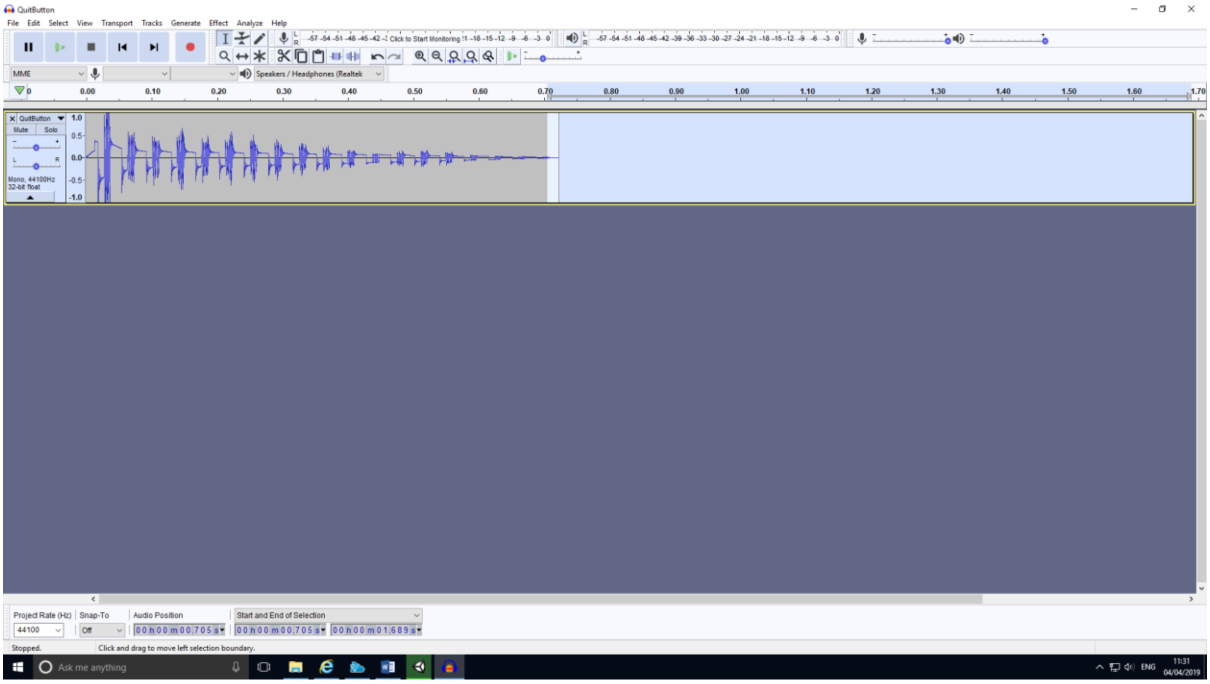


Figure 6: After Effects, shorten matrix video

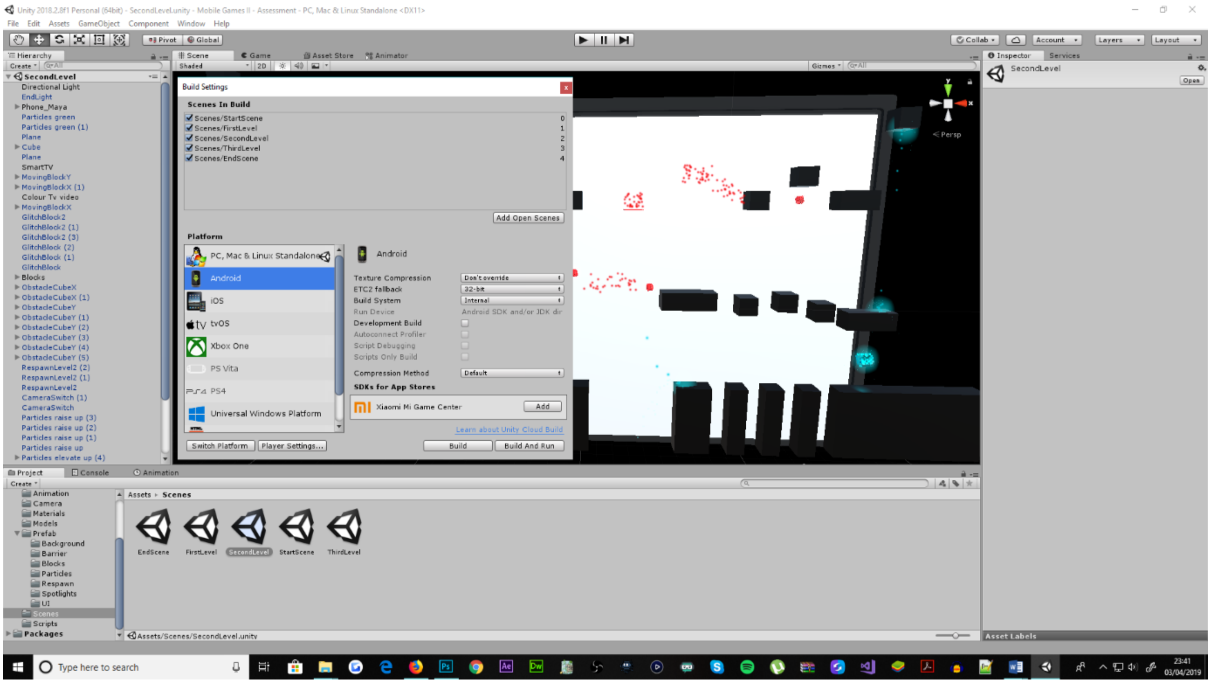

QuitButton
File Edit Select View Transport Tracks Generate Effect Analyze Help
MME
Speakers / Headphones (Realtek
QuitButton
Mute Solo
Mono, 44100Hz
32-bit float

Project Rate (Hz): 44100
Snap-To: Off
Audio Position: 00 h 00 m 00.705 s
Start and End of Selection: 00 h 00 m 00.705 s – 00 h 00 m 01.689 s

Stopped.

Click and drag to move left selection boundary.

Figure 7: Audacity, clipping audio length


Unity 2018.2.8f1 Personal (64bit) - SecondLevel.unity - Mobile Games II - Assessment - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
Scene
Game
Asset Store
Animator
Inspector
Services
SecondLevel
Project
Console
Animation
Assets ▸ Scenes
EndScene
FirstLevel
SecondLevel
StartScene
ThirdLevel
Assets/Scenes/SecondLevel.unity

Build Settings

Scenes In Build

- ☑ Scenes/StartScene 0
- ☑ Scenes/FirstLevel 1
- ☑ Scenes/SecondLevel 2
- ☑ Scenes/ThirdLevel 3
- ☑ Scenes/EndScene 4

Add Open Scenes

Platform: PC, Mac & Linux Standalone; [✓ Android]; iOS; tvOS; Xbox One; PS Vita; PS4; Universal Windows Platform

Android

Texture Compression: Don't override
ETC2 fallback: 32-bit
Build System: Internal
Run Device: Android SDK and/or JDK dir
Development Build: □
Autoconnect Profiler: □
Script Debugging: □
Scripts Only Build: □
Compression Method: Default

SDKs for App Stores: Xiaomi Mi Game Center — Add

Learn about Unity Cloud Build

Switch Platform | Player Settings... | Build | Build And Run

Figure 8: Building executable file (.apk)