Simplified Dynamic Scenarios Documentation

1. General Project Description

Simplified Dynamic Scenarios (SDS) consists of a dashboard with an embedded QR Scanner. That scanner will proceed to place in world space whichever of the supported content types as 3D objects.

Supported content types are:

1. <u>Images</u>: png, tiff, jpeg, jpg, gif
2. <u>Video</u>: mp4, YouTube (perhaps include only iframe?)
3. <u>IP Camera Livestream</u>: IP URL format (e.g. 127.0.0.1), mjpg, mpg
4. <u>3D Objects</u>: obj, zip(containing obj & mtl)

2. Development Package Versions

- <u>Unity</u>: 2019.3.5f.1 Personal
- <u>Vuforia Core Samples</u>: 9.2.1 (9.0.0)
- <u>Unity Samples UI</u>: 1.2.2 (1.2.1)
- <u>Standard Assets</u>: 1.1.5
- <u>TextMesh Pro</u>: 1.2.2
- <u>Runtime OBJ Importer</u>: 2.2.0

3. Target Platforms

- <u>Android</u>:
  - Minimum API Level: 4.4 (KitKat)
  - Note: Graphics → Always Included Shaders → Standard (Specular Setup)
- <u>Microsoft HoloLens</u>:
  - Pending

4. Scene Layout

- An AR Camera is placed in the middle of the scene.
- Two separate light sources, facing both east and west.
- A UI Canvas, containing most basic project components.
  - → Target: blue square, scanning area (image)
  - → Status: Processing button, only visible on data load

→ Close: X button, removes scanned object from scene (destroys GameObject from scene)
→ DisplayImage: Contains image projection after scan
→ Zoom: Slider for image zoom
→ Video: contains VideoPlayer, used to stream videos
→ Play/Pause: buttons for respective function calls while video streaming
→ MeshGenereator: ???

- UserDefinedTarget + UserDefinedTargetBuilder: created in order to be able to produce a UDT at runtime on button click (see below)
- TargetBuilderUI: contains a transparent BuildButton. This button contains a UserDefinedTargetBuilder function call onClick, called BuildNewTarget. This functionality will be discussed further on. In practice, this button once an item is scanned is automatically invoked to build a new target for every item.
- CommonUI: ????

5. Functionality

This application's functionality consists of scanning a multitude of different items and placing them in world space. Such items may be images, videos, objects or streams.

Once the app starts running, every basic component is set in its respective mode and a Vuforia instance is created. From this point on, Vuforia will update continuously until a suitable trackable is located. Camera formats and focus are set and the barcode scanner is configured.

Note: If an object is set and the given url is a Dropbox or a Google Drive link, the correct format is ensured. After that, the content is loaded from Web.

LoadFromWeb() identifies every content type and functions accordingly in each case:

- In case a YouTube url is scanned, a system executable is called to run automatically to download the content as mp4 and preview as such. Once identified, the routine for regular video files is invoked.
- An IP url is saved as a backup and replaced by an intentional url. This serves as an identifier for later to create a unique case for IP urls.
- A Google Drive url is replaced by its export equivalent. This happens due to Google's policy not allowing a preview of its content (header's content returns bytes). This solution simply exports the content and previews the local file when needed.
- If none of the above occur, a web request returns information about the content type and allows the function to continue accordingly.
- If no errors occur, all cases are split, in order to present the suitable information for each case.

- IMAGE: Image supported data types: tiff, jpeg, jpg, png, gif. If no other image is currently displaying, implementation begins. First step is to create a 3d plane and scale it to image size. The image is created as a sprite, and its materials and textures are loaded into the plane, which is renamed as “object”, a recurring motif throughout this project. Also, the close button and the slider are activated.

  Once everything is set, the BuildButton is invoked and BuildNewTarget() is called.

- VIDEO: Video supported types: mp4. Close Button is activated and a routine for Video playing is invoked. This routine sets the audio source and waits until the video player is free to acquire. When this occurs, a plane is created in the same way as described above, and materials and textures are copied into the plane, which is again renamed as “object”.

  Afterwards, the url-provided video is passed onto the video player, inside the scene, buttons and audio sources are enabled and BuildButton is invoked again.

- IP: An IP stream is recognized as the “decoy” url https://this-page-intentionally-left-blank.org/, in which case the original IP is copied back into place. Using a slightly differentiated version of the implementation described here: https://answers.unity.com/questions/1151512/show-video-from-ip-camera-source.html , which basically loads the stream into the video player. (EXPLAIN MORE)
- ZIP: Accepted when containing .obj and .mtl. After identifying and extracting the file, the zip file is deleted ad the object is renamed along with its material. If no other 3d object is currently displaying, the object is loaded using this runtime object loader: https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547 and yet again, invoking the BuildButton.
- 3D OBJECTS: currently only .obj supported (ADD FBX AT SOME POINT). Similar to zip file’s implementation, only in this case, no material is loaded into OBJ Runtime Loader, but instead a default material is used.
- ELSE: Simply skips. Add an “Unknown file type, cannot preview” prompt.

BuildNewTarget(): Vuforia’s core samples provide a very useful function, in which, depending on frame quality (ADD INFO ABOUT TOGGLE FLASH) a new target is created inside the scene, depending on the clearest object of the camera’s surroundings. This function is what the implementation of transferring data to 3d is based on, since the object will be placed atop the new target. Possible cases:

- .obj: if such a file exists, replace its previous position with coordinates of the center of current scene.
- .zip: if an object directory exists, work as in .obj. In both cases, mark a flag as 1. Alternative means image or video needs to be placed in 3d space.
- flag=0/1: different scaling and rotating techniques, same process.

If the target is not clear enough, a message is displayed by default to let the user know that the camera quality is low.

NOTE: In order to achieve maximum results during runtime, the camera view needs to be as clear as possible. For this reason, once the camera is instantiated, its flash torch mode is set to active.

6. Sources

1: https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547

2: https://answers.unity.com/questions/1151512/show-video-from-ip-camera-source.html

3: https://www.youtube.com/watch?v=oklcNZpx1-U

4: https://github.com/ytdl-org/youtube-dl

5: https://this-page-intentionally-left-blank.org/

6: https://answers.unity.com/questions/1370621/using-videoplayer-to-stream-a-video-from-a-website.html