MovieFX: Combining the real and the virtual

Part 1

Petr Schreiber

Introduction

Welcome to the new MovieFX series of TBGL tutorials! As the name suggests, they will introduce you to various ThinBASIC techniques, which can be used to produce moving pictures.

The "Combining the real and the virtual" sub branch of the movie related tutorials focuses on one extremely attractive topic of today, which is enhancing pictures and movies of real world around us with computer generated elements.

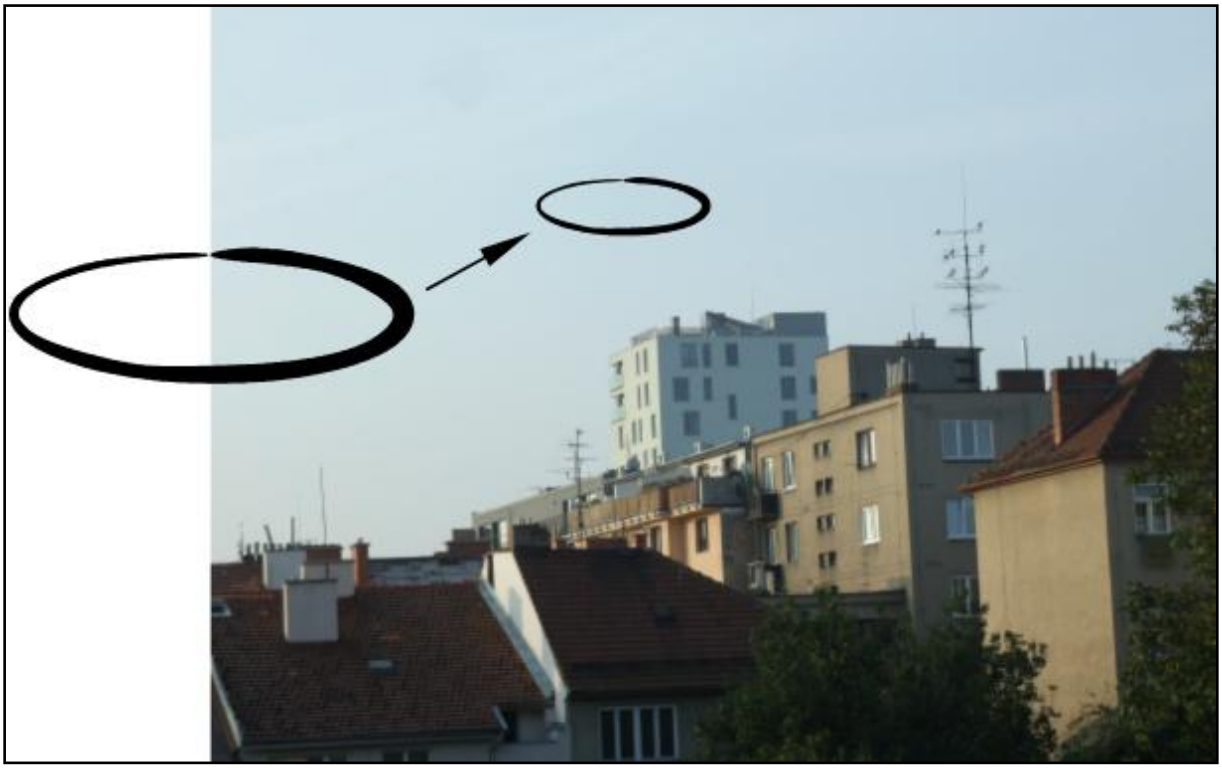
The structure of tutorials is done in classic way from basics to more complex topics.

The series “for advanced” presume the reader is already familiar with TBGL .

MovieFX series start with very simple scene, which will show you how to turn still photo into less static video with one CG object added.

Planned clip

Our goal for this lesson will be to take still photo from digital camera, and add Unidentified Flying Object to it. The script will produce series of bitmaps, representing the flight of the UFO, which we will convert to AVI file using freeware VirtualDub[1].



Pic. 1 Planned scene - UFO flying from the observer out to the distance

[1] You can download this tool at http://www.virtualdub.org/

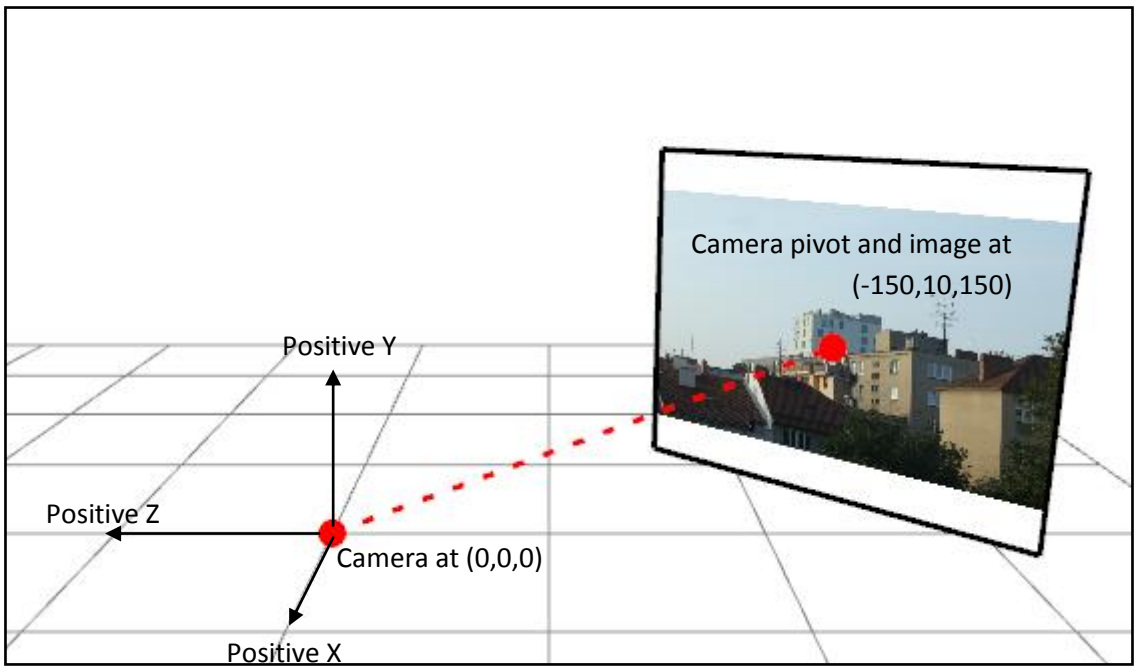
Setting up the scene

We will use TBGL module and Scene-Entity system to accomplish the job. Our scene will consist from few basic elements:

- camera and its target pivot
- background photo
- UFO model
- light

Camera is essential, as it represents our view on the scene. The white house in the centre of the photo is in reality approximately 200 meters away from the digital camera, and we are pointing up (and also sideways) with the camera, so it marks place 10 meters above the observer on the wall of white house.

The following picture explains the situation:



Pic. 2 Position of camera and its pivot

We will place the virtual camera at position (0, 0, 0) and we will point it to the pivot point (150, 10, 150), which is the point at the wall of the house we looked on in the reality.

The pivot of the camera is used to enable one very important thing – shaking of the camera. As we use still image and 3D model, to add the sensation of watching video captured by camera in hand it is enough to shake the pivot point. This way we will get synced shaking of background and other 3D objects added to the scene.

Background photo has been shot with digital camera. It has the 4:3 proportions. As you might remember, older 3D cards do support only resolutions of power of two, so you should resample the image to 1024x768, and add 128 pixels to top and bottom, to make it squared and power of two.

The quad with background image should be scaled in a way it is not completely visible on the screen. This is just to make possible shaking of the image. In case the image would be scaled to fit the frame, the borders of the quad would be visible during the shaking which would completely ruin the effect.

To make image always face camera, just use *TBGL_EntitySetTarget* to make your background entity look directly at the camera. To prevent interaction of light and fog with the frame, use *TBGL_PushStateProtect* / *TBGL_PopStateProtect* commands to isolate the rendering of the texture quad.

UFO model can be created using any 3D modeller with support for OBJ export. You can use *OBJ to M15* tool to convert the file to TBGL friendly M15 file format.

You can animate the UFO any way you like, for example using *TBGL_EntityPush* and *TBGL_EntityTurn* commands.

Simulating light and atmospheric conditions

The photo has been shot at very special light conditions - on the morning, when the front, bright face of the white house is directly illuminated by Sun, and the other side is basically not lit. It is essential to bring these light conditions to our virtual space as well.

Light in TBGL system has 2 colour related properties - diffuse colour and ambient colour.

The ambient colour is present on faces not lit by light, while the directly lit faces are coloured using sum of RGB components of diffuse and ambient light colours.

Using your favourite bitmap editor, you can spot the bright face has colour approximately 208,214,193 and the dark side of the house has colour of 100,126,138.

Note: We are lucky the house is white. In other situations it would be advantageous to prepare simple white cube painted with matt white colour to use as reference.

So we can set the ambient part of the light like:

```
TBGL_EntitySetAmbient(%sScene, %eLight, 100, 126, 138)
```

The final colour of the light on fully illuminated face is sum of the diffuse and ambient parts. That means, if colour of the bright wall of the house was 208,214,193 and the dark 100,126,138, then the diffuse component of the light equals their difference.

```
TBGL_EntitySetColor(%sScene, %eLight, 208-100, 214-126, 193-138)
```

Pic. 3 Places where you should take the color samples

As the UFO will be truly "alien" element to the scene (2D background vs. 3D object), it is good to somehow improve the feeling of the depth. When you stand on high mountain, you can see the terrain in distance fades away in some kind of fog, even at clear weather. This atmospheric effect will be our key to improve the depth feel.

We will simulate this using TBGL fog. As the colour of the fog, we will use colour of the sky around the houses. It can be for example 206,224,229. Just keep in mind to setup the fog very thin and subtle, to not hide UFO in the atmosphere too soon.

Generating bitmaps

To finally generate the video, we need to export series of bitmaps first. This can be achieved using *TBGL_SaveScreenshot* command. Simply use counter variable to create file names with ascending numbering. It will save you work of manual arrangement of files later.

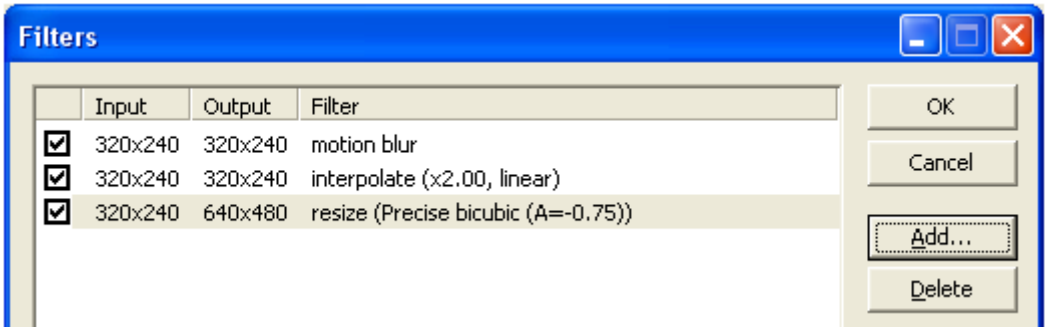
As we are targeting video, we want quality. To avoid rendering of teared frames, always enable vertical synchronization using *TBGL_UseVsync*. This ensures the screen will be updated only when whole frame is ready.

Another very important note is to use manually specified constant frame rate when using the script in frame generation process and do not use any functions dependant on time, such as *timer*, *getTickCount* or *hiResTimer_Get*.

Final touch

Once you generate numbered frames of the animation, start VirtualDub and drag the first file in series to the program window, others will be added automatically.

Then you simply select target frame rate using Video/FrameRate option, and add suitable post processing filters. I recommend adding motion blur, interpolation and down sampling the frames to target resolution[2]. The filter dialog then might look like on the picture.



Pic. 4 Used filters in Virtual Dub

We could do the motion blur and interpolation in ThinBASIC, but it would make no point when the VirtualDub does the job for us. This way we can keep the script relatively simple, and do the routine work using third party tool.

Frame from final rendering can look similar to the image below.

Pic. 5 Sample rendering

You can observe the lighting and blur caused by shake are in sync. The fog fading to atmosphere colour starts to manifest at this distance, helping to better integrate the UFO.

[2] It is good idea to render images at higher resolution than the target one is, down sampling in Virtual Dub will smear jagged edges of 3D objects, in case you did not used antialiasing