Animation Final Project Report
Animated Short Film: The Boy and the Magician

Xue Yu (Zoe YUU), Trinity College Dublin, MSc. Computer Science – R and VR

Introduction

This project has been inspired by the interlude of a Chinese fanfiction named *Good Luck*, featuring a lost boy and his encounter with a magician. Techniques such as timelines, cinemachine, physics, (attempt of faked) soft body, particles system, simple AI (chase/follow), IK, motion capture, facial expression and (faked) lip synchronization have been used to tell the story. The project has been implemented using Unity 2017.4 for animation, Blender for shape keys generated animation keyframes and Motionbuilder for cleaning the motion data and retargeting to a character, 3D characters and most animations used are generated and downloaded from Mixamo. The actress of the captured motion data was me.

Content of the projects and the general storyline

The entire story consists of two Unity projects[1]: GL1 and GL2. GL1 contains 6 scenes, which have been connected with each other by keyboard event[2].

Scene 0, "Chase and lost": (The boy who has lost his parents and wandered to the icy-cold land of Siberia, struggling to stay alive by working for a circus as the caretaker of small animals. However, after a chaotic event, all animals ran out. A couple of penguins are the last animals the boy still kept in the sight and had hope to chase back.) The boy was chasing the last penguin on the icy ground, however, as they ran close to three circus tents, weird balls were bouncing around near the circus tents, which had caused the boy to lose sight of the last penguin.

Scene 1, "Stranger in Snow": Sinking into despair, the boy was walking back alone, with nowhere to return to and basically lost every chance or willpower to survive. The snow began to fall. The boy looked up at the snow, completely lost and feeling defeated.

Scene 2, "the Magician": whereas the boy was standing in defeat, from far away and nowhere, a magician came, stopped in front him. The boy barely acknowledged his presence. However, the magician was loud and stating that the boy has stolen his candy. Even in his despair, the boy refused. The magician said angrily, "But I can see them, in your right pocket!", the boy shook his head again.

Scene 3, "Flowers in Winter": Suddenly, flowers appeared from the boy's right side, flowing through the air above his head; the boy was amazed and startled at the same time, the magician laughed and said, "If not in your right pocket, then, in your left pocket!" The boy shook his head again, but again, suddenly, stars appeared from his left side, flowing through the air. The boy was amazed.

---

[1] Due to the limited computation resource of the computer used to develop this project.

[2] Press key space, enter the next scene, the script controls this is SwapScene.cs, added on an empty GameObject "SceneManager" in every scene of GL1.

(The boy felt like that he was living in a fairy tale: the right hand defeated the winter by making flowers blooming; the left hand put stars in the sky to lighten up the darkness. The children in the whole world would smile and laugh at those magical tricks.)

Scene 4, "the magician dances": Grinning, the magician danced and applied his last magic: a cylinder hat full of candy has flown to the boy.

Scene 5, "a Hat of Candy": the hat flew down into the boy's arm, the boy received it with both hands.

In the single scene of GL2, a smile appeared on the boy's lips.

Implementation and technical details

The projects have been implemented on a Surface 3, with Win10 as operation system, the Intel(R) Atom(TM) x7-Z8700 as CPU (1.60 GHz), and Intel(R) HD Graphics as GPU.

The scenes all consist of a gray plane which resembles the icy land of Siberia, a directional light has been rotated to a certain angle so that the environment resembles the illumination of a depressing twilight. In project GL1, each scene contains an empty GameObject "SceneManager" with script "SwapScene.cs"[1], the viewer can press key space to proceed to the next scene.

Code/Pseudocode can be found in footnotes.

GL1

Scene 0

Characters: the boy is the 3D character "Ty" with animations downloaded from Mixamo; the Penguin and the tents are the free model[2] with animation downloaded from Unity assets shop.

Two cinemchine virtual cameras have been added to the scene: Cam1 follows the penguin but looks at the boy from a third person perspective, so that the viewer can see that a chase is going on; Cam2 follows the boy from the viewpoint behind him, so that the viewer can see clearly what happened when they come close to the tents. Press key 1/2 can swap camera view in ease-in-ease-out blend method[3].

[1] In the Update() in the script, SceneManager.LoadScene(SceneIndex); will be called, to load the next scene.

[2] Imperial Penguin: https://assetstore.unity.com/packages/3d/characters/animals/imperial-penguin-100397
Medieval Tent Big: https://assetstore.unity.com/packages/3d/environments/historic/medieval-tent-big-19023

[3] Transition time: 1s. The controlling script is "SwapCam.cs".
```
if (Input.GetKeyDown (KeyCode.Alpha1)) {Cam1.SetActive(true); Cam2.SetActive(false); }
```

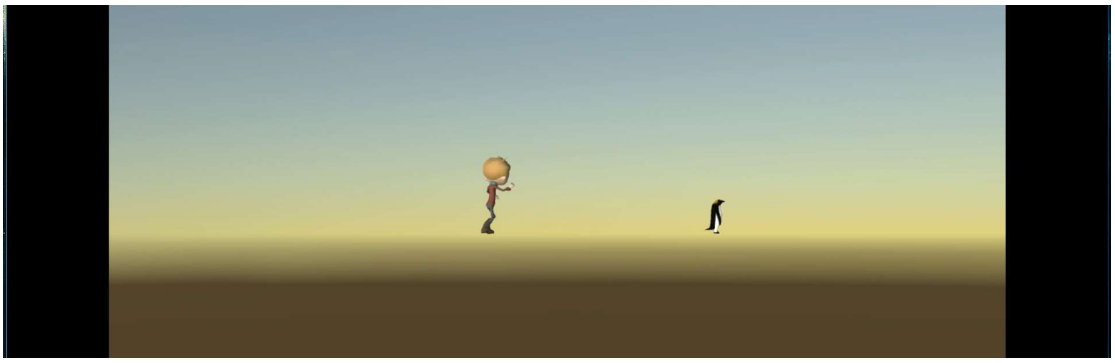


Fig. 1. view from camera1

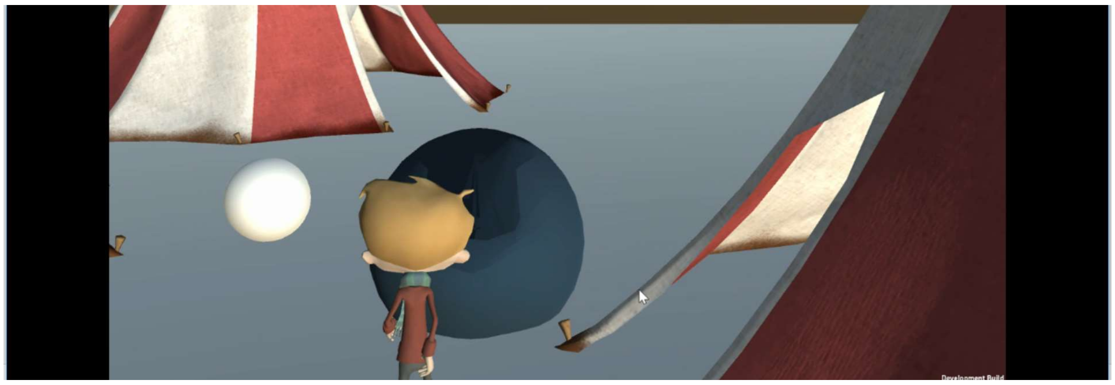
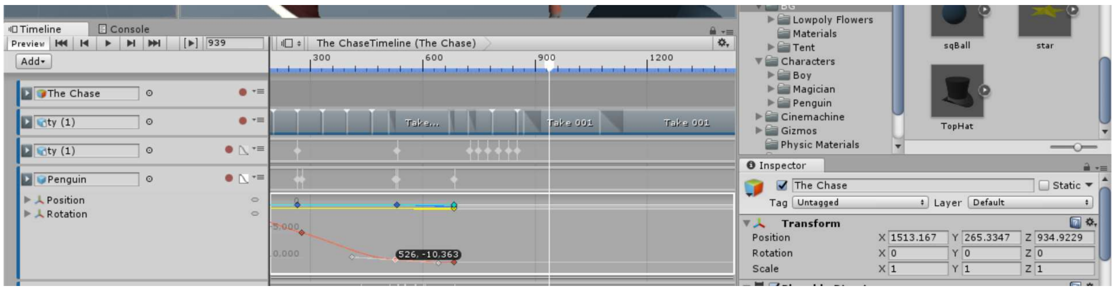


Fig. 2. view from camera2

Timelines of the boy, the penguin and the squeeze ball have been added to an empty GameObject “The Chase”.

For the boy, 5 animation clips have been used: running- with hands waving around in the air in panic, stop, surprised, look around, sigh in anger/defeat. Efforts have been given to adjust the position of those clips and the speed of the clips so that the transition between gestures can be as smooth and natural as possible. Also, timeline function of Unity 2017 allows operations such as to set animation tracks alongside the added animation clips, so that the character’s position and rotation in x/y/z 3 dimensions can be recorded into keyframes and fine-tuned (for instance the gradient of the curve resembling the x/y/z axis translation can be manually adjusted, so that the movement of the character of this axis could remain a steady speed)



The Chase
Tag Untagged Layer Default
Transform

```
Position   X 1513.167   Y 265.3347   Z 934.9229
Rotation   X 0          Y 0          Z 0
Scale      X 1          Y 1          Z 1
```

Fig. 3. example of fine-tune the gradient of the rotation-curve; adjust of transition and speed of animation clips

The penguin only has one animation clip: running, its body rotates a little during the chasing, first rotates to observe the boy, then looking ahead, heading for a hiding place[1].

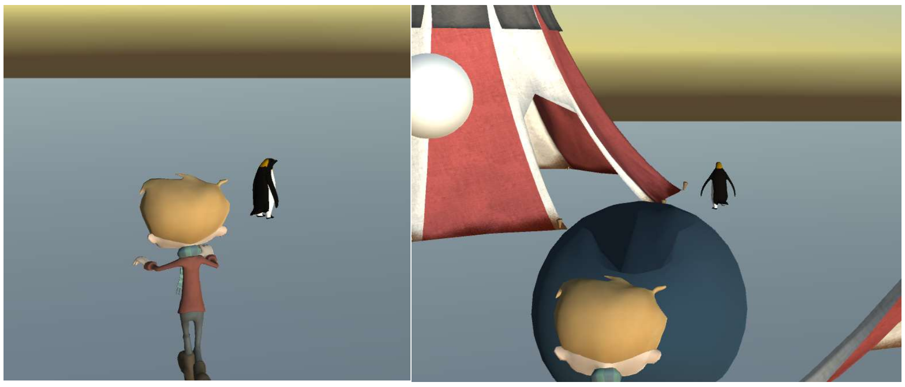
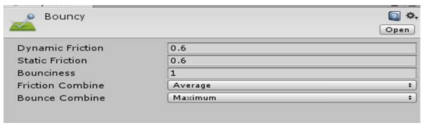


Fig. 4. the rotation of the penguin

Two balls, especially the squeeze ball has impeded the boy's chasing of the penguin. The squeeze ball is a poor attempt made to resemble a soft body effect on a ball. Ideally, a soft body flag anchored on a tent's door should be the object which impedes the boy's chasing, in which it is to be blown up by a wind[2] and stops the boy's forward movement. However, as it turns out with Unity 2017.4, vertices for constraint cannot be selected/painted and the attempt of using the proper soft body failed. Since the boy needs a distraction to get his attention from the penguin – an object modelled in Blender has then been added in the scene, which contains animation keyframes of been squeezed which have been generated with a series of shape keys. As the boy came close to the tent, the squeeze ball's timeline makes it fall to the ground and bounces a little.
Another white ball has been added in the place near the tent, with the component of rigid body and a material called "bouncy" added to it. As the scene begins, this ball will fall from above and keeps bouncing around the boy, as he looks helpless around.


Bouncy

Open

Dynamic Friction	0.6
Static Friction	0.6
Bounciness	1
Friction Combine	Average
Bounce Combine	Maximum

Fig. 5. material Bouncy's parameter

Animation principles demonstrated in this scene:
Squash and stretch: demonstrated with the faked soft body squeeze ball.
Ease in, ease out: the blend of the cameras, the blend and adjust of the movements of the boy.
Arcs: the boy run around the squeeze ball; the penguin run around the tent.
Anticipation: the blend and adjust of the animation clips of the boy.
Exaggeration: the boy's panicking running gesture, looking around, throw up hands in anger and defeat.
Interactive element: swap of cameras and the scene.

[1] Behind one of the tent.
[2] Horizontal force given by a key pressed by the viewer.

Scene 1

Character: the boy

Cameras: 2 cinemachine cameras have been added: the first one shows the boy walking alone defeatedly in the snow; the second looking down from above to capture the boy's face, as he stops, looks at the sky (in exaggerated anger and despair). Press 1/2 swap cameras in ease-in-ease-out blend method; the script "SwapCam.cs" has been added to the camera object.

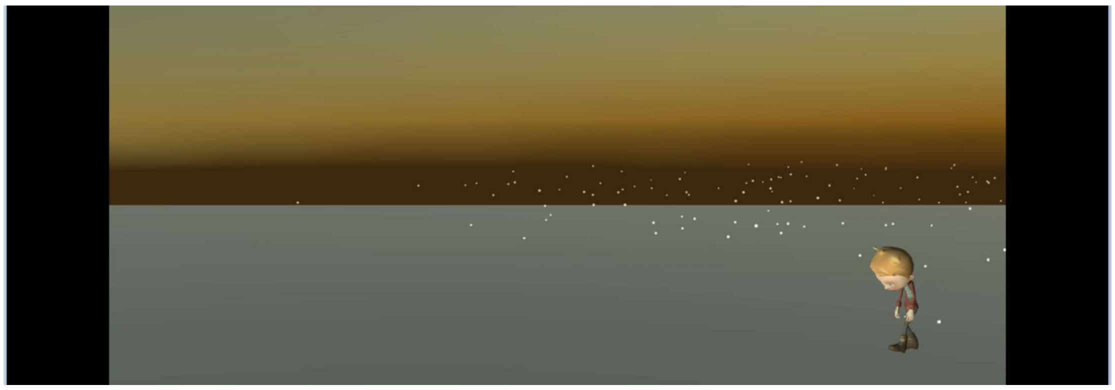


Fig. 6. view from camera1

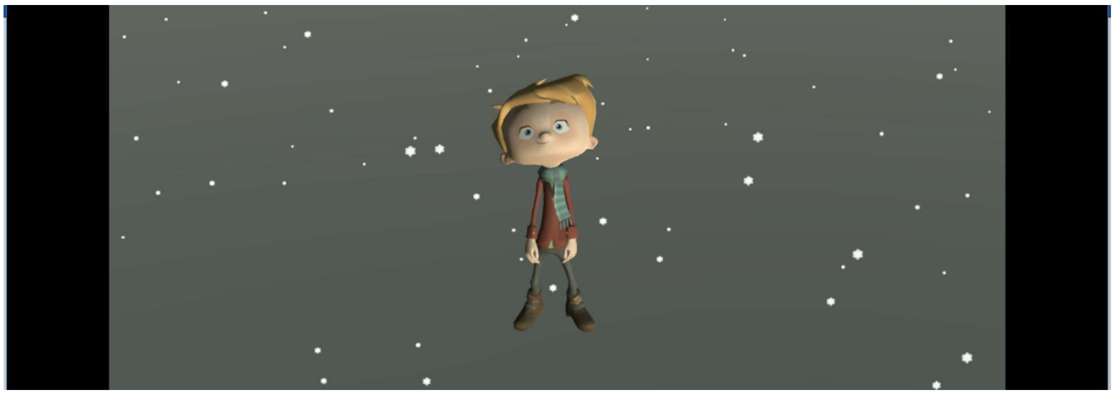
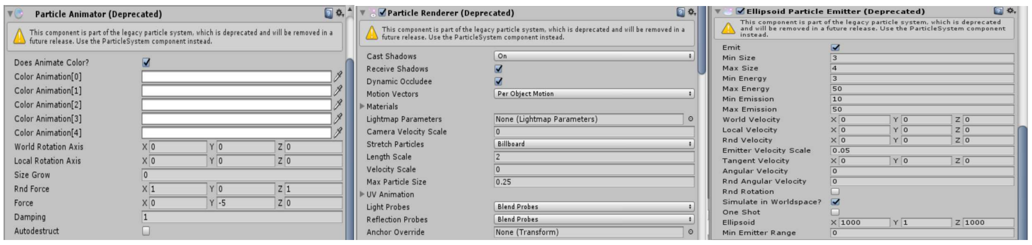


Fig. 7. view from camera2

Timelines of the boy have been added on an empty GameObject named "Timeline_StrangerInSnow".

The snow (particles) has been implemented using 3 components: Particle Animator, Particle Renderer and Particle Emitter[1].



```
Particle Animator (Deprecated)
This component is part of the legacy particle system, which is deprecated and will be removed in a
future release. Use the ParticleSystem component instead.
Does Animate Color?     ✓
Color Animation[0]
Color Animation[1]
Color Animation[2]
Color Animation[3]
Color Animation[4]
World Rotation Axis     X 0    Y 0    Z 0
Local Rotation Axis     X 0    Y 0    Z 0
Size Grow               0
Rnd Force               X 1    Y 0    Z 1
Force                   X 0    Y -5   Z 0
Damping                 1
Autodestruct            ☐
```

```
Particle Renderer (Deprecated)
This component is part of the legacy particle system, which is deprecated and will be removed in a
future release. Use the ParticleSystem component instead.
Cast Shadows            On
Receive Shadows         ✓
Dynamic Occludee        ✓
Motion Vectors          Per Object Motion
Materials
Lightmap Parameters     None (Lightmap Parameters)
Camera Velocity Scale   0
Stretch Particles       Billboard
Length Scale            2
Velocity Scale          0
Max Particle Size       0.25
UV Animation
Light Probes            Blend Probes
Reflection Probes       Blend Probes
Anchor Override         None (Transform)
```

```
Ellipsoid Particle Emitter (Deprecated)
This component is part of the legacy particle system, which is deprecated
and will be removed in a future release. Use the ParticleSystem component
instead.
Emit                      ✓
Min Size                  3
Max Size                  4
Min Energy                3
Max Energy                50
Min Emission              10
Max Emission              50
World Velocity            X 0     Y 0    Z 0
Local Velocity            X 0     Y 0    Z 0
Rnd Velocity              X 0     Y 0    Z 0
Emitter Velocity Scale    0.05
Tangent Velocity          X 0     Y 0    Z 0
Angular Velocity          0
Rnd Angular Velocity      0
Rnd Rotation              ☐
Simulate in Worldspace?   ✓
One Shot                  ☐
Ellipsoid                 X 1000  Y 1    Z 1000
Min Emitter Range         0
```

Fig. 8. the components of the particles/snow

[1] Because this approach is the same with the one used in OpenGL when building a particle system for snow simulation.

4 Animation clips of the boy have been used in this scene: the sad walking, raise head – which is a clip cut from the clip "surprise"[1], defeat and sad idle. The gestures of the boy have been fine-tuned so that the movement looks natural. For instance, the last clip, "sad idle" has been played at half speed so that it can be merged with the previous clip "defeat" smoothly.

Animation principles demonstrated in this scene:
Ease in, ease out: the blend of the cameras, the blend and adjust of the movements of the boy.
Arcs: the moving track of the camera.
Anticipation: the blend and adjust of the animation clips of the boy.
Exaggeration: the boy's walking (head hanging), looking up, throw up hands in anger and defeat.
Interactive element: swap of cameras and the scene.

Scene 2
Characters: the boy; the magician is the 3D character "whiteclown_n_halin", character and animations download from Mixamo.
Cameras: 2 cinemachine cameras have been added: the first camera captures the magician's gestures and movements; the second captures the boy's gestures and movements. Press 1/2 to swap the camera in "cut" blend method to capture the characters' quick interaction.

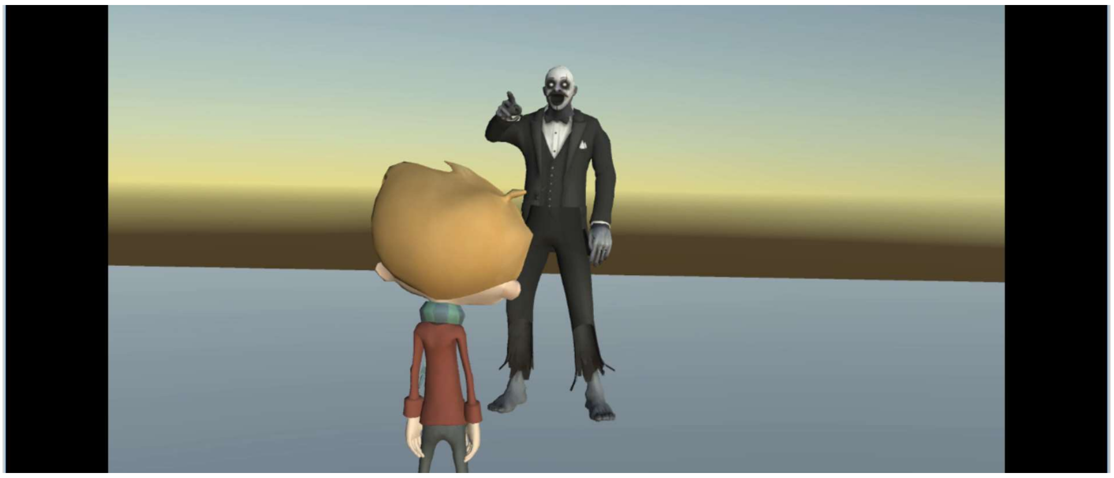


Fig. 9. view from camera1

[1] By selecting the start keyframe, stop keyframe and adjusting the range of the movement directly on Mixamo before downloading the animation.

Fig. 10. view from camera2

Timelines have been added to the empty GameObject “The Magician Timeline”.

5 Animation clips of the boy have been used and blended together: sad idle, raise head, shaking head, step back – which is a clip generated from the reserved walking[1], surprise.
4 Animation clips of the magician has been used and blended together: walking, stop, pointing, dismissing.

Animation principles demonstrated in this scene:
Ease in, ease out: the blend and adjust of the movements of the boy.
Anticipation: the blend and adjust of the animation clips of the characters.
Exaggeration: the magician’s accusing hand gesture; the boy’s startled gesture and stepping back gesture.
Interactive element: swap of cameras and the scene.

Scene 3
Characters: the boy and the magician
Cameras: 3 cinemachine cameras have been added: the first captures the boy’s face as he looks up as flowers flying – for in this scene the movement of the flowers and the stars are the focused points, the second captures the magician’s movement, the third captures the boy’s movement and the flying of the star. Press 1/2/3 swaps cameras in “cut” blend method in to capture the fast interaction between the characters.

[1] Generated directly on Mixamo in the same way which has been used to generate the “look up” gesture.

Timelines have been added to the empty GameObject "Flowers in Winter".

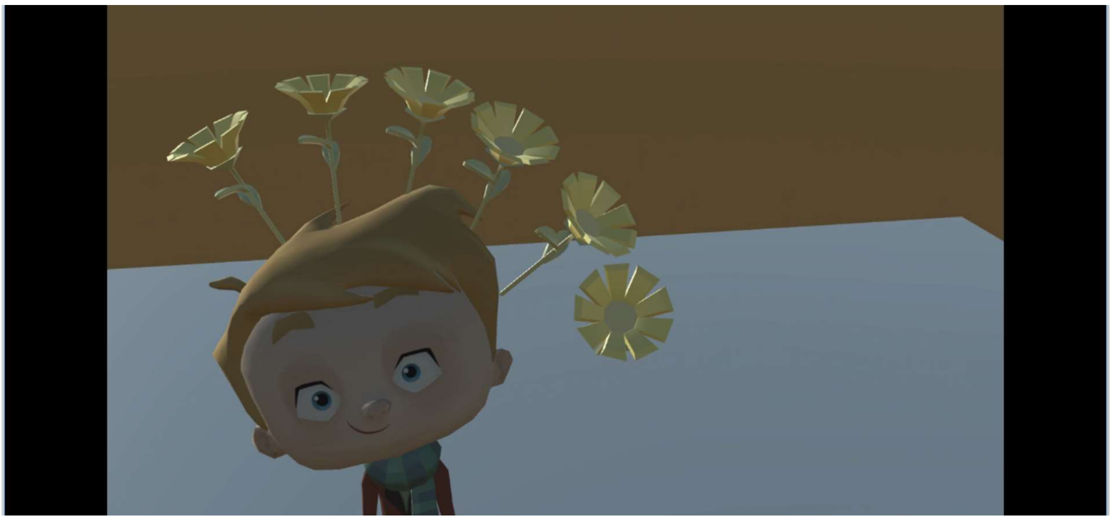


Fig. 11. view from camera1

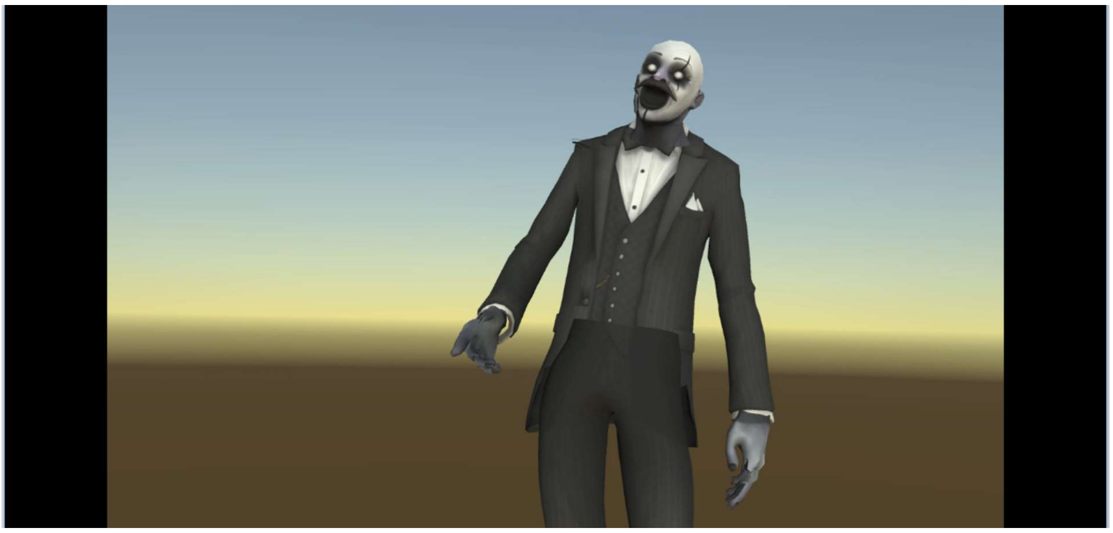


Fig. 12. view from camera2

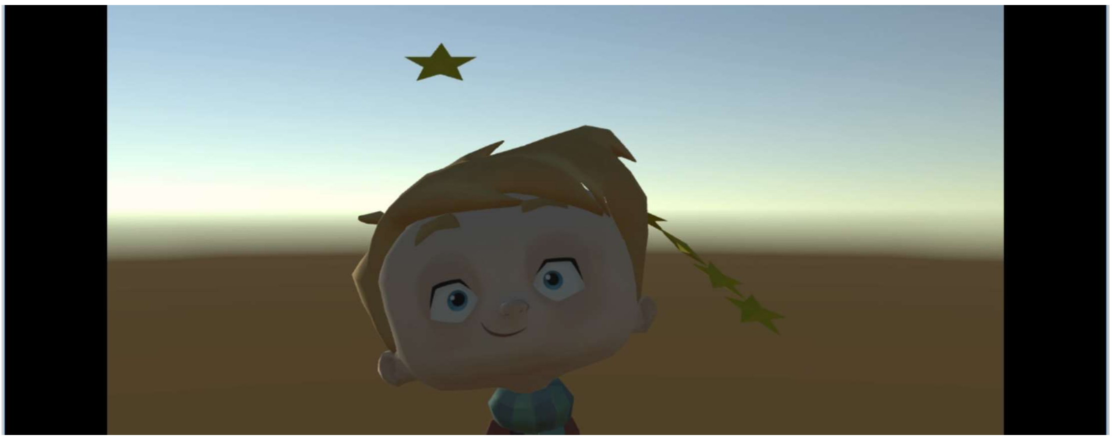
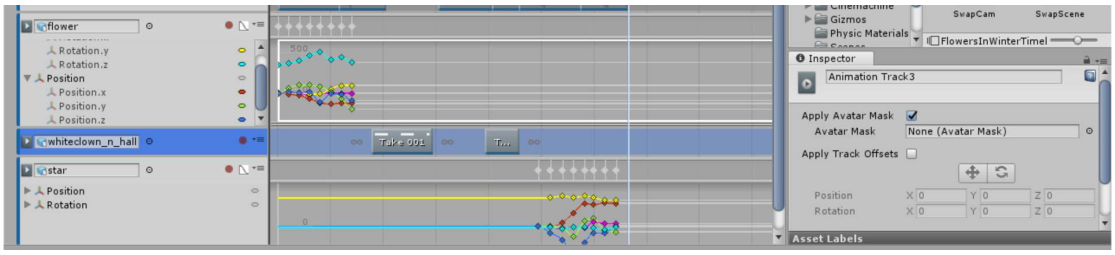


Fig. 13. view from camera3

Timelines have been added to the empty GameObject "Flowers in Winter".

The movements of the flowers[1] and the stars[2]:

The simplest AI script "FollowAI.cs" has been added on all flowers[3] and stars except the first of each, which has their respective timelines – the first flower flies from the boy's right to left, whereas the first star flies from left to right. The flying tracks of the first flower and the first stars have been carefully set in their respective timelines.



Animation Track3

Apply Avatar Mask ✓
Avatar Mask None (Avatar Mask)
Apply Track Offsets ☐

Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0

Fig. 14. complicated movement tracks for the first flower and the first star

Animation principles demonstrated in this scene:

Ease in, ease out: the blend and adjust of the movements of the boy.

Arcs: the moving tracks of the flowers and the stars.

Anticipation: the blend and adjust of the animation clips of the characters.

Exaggeration: the boy's head following the movement of the flowers and the stars; the magician's hand gestures.

Interactive element: swap of cameras and the scene.

Scene 4

The motion capture data[4] has been cleaned in Motionbuilder and retargeted to a character "Gremlin"[5] in Motionbuilder's samples. The cylinder hat[6], which resembles the hat full of candy in the story, has been attached to the Gremlin/magician's right shoulder.[7]

---

[1] Free model Lowpoly Flowers from Unity Asset: https://assetstore.unity.com/packages/3d/vegetation/plants/lowpoly-flowers-47083

[2] Free model Star from Free3D: https://free3d.com/3d-model/star-mobile-ready-60-tris-49986.html

[3]
```
transform.LookAt(Target);
 if (Vector3.Distance (transform.position, Target.position) >= 25)
   {transform.Translate(Vector3.forward * 100 * Time.deltaTime);}
```

[4] With the reporter herself as the actor. The motion captured is aimed to act as if the character is performing a magic trick.

[5] An attempt has been made to characterize the magician's 3D character model "whiteclown_n_halin" but failed. In order to imply the movement of the magician using magic on a character, the build-in character "Gremlin", which is most closely resembles a magic being, has been chosen to play the role of the magician. However, as tried to export the animated textured character through "file", only the eyes of Gremlin have been exported. Saving the character directly in fbx file resulted in a Gremlin 3D character with animation but without texture.

On the other hand, the brown furry texture of Gremlin only makes this creator less human and resembles the magician even less. Thus, no further attempts have been made to add texture to it.

Although a Gremlin is not a magician, since in the original story the boy felt like as if he was living in a fairy tale – it could be interpreted that the boy imagined that the human magician actually is like a Gremlin in the fairy tale.

[6] Free 3D model from Sketchfab, https://sketchfab.com/models/e3c113c291c24088bb2559b39d48b778?ref=related

[7] Although the hat also has its own timeline to adjust some of its rotation angle.

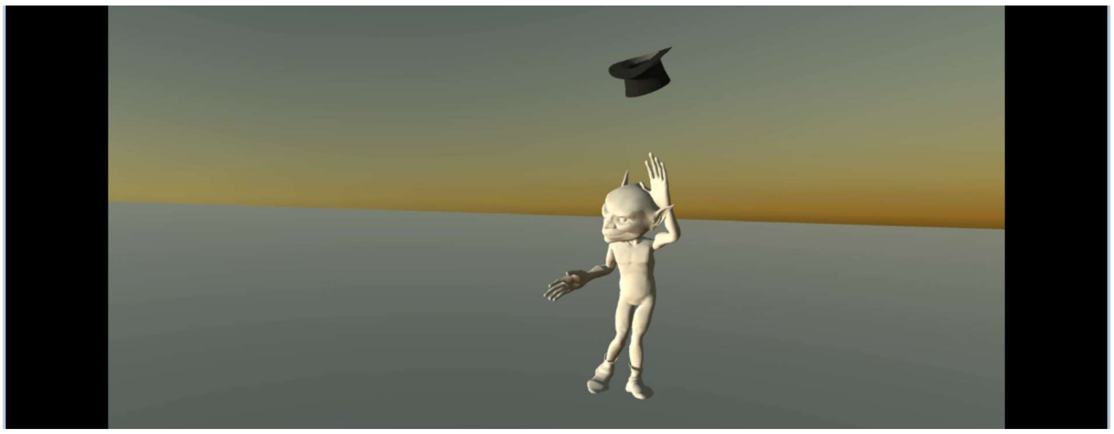


Fig. 15. Gremlin/magician does his magic trick with top hat attached to his shoulder as child

Animation principles demonstrated in this scene:
Arcs: the movement of the cylinder hat.
Exaggeration: the movement of the Gremlin/magician as he did magic tricks; the flying track of the cylinder hat.
Interactive element: swap of the scene.

Scene 5

Character: the boy.
The hat has the timeline in which it is flying down from the sky. With Ik has been set active, an animator controller and a simple IK script "ReceiveIK.cs" has been applied to the boy, in which both hands are reaching out to the position of the hat and the boy look at the hat[1].

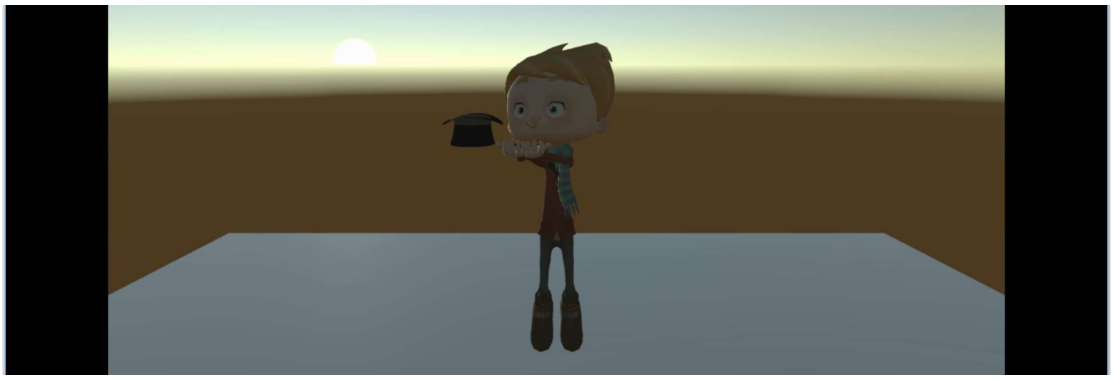


Fig. 16. the boy receiving the hat, by watching the hat and both hands reaching out to the hat

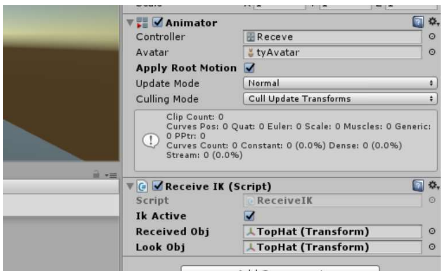
[1] For the LookAt move:

```
            animator.SetLookAtWeight(1);
            animator.SetLookAtPosition(lookObj.position);
```

For movements of the right hand (code with the left and is the same):

```
            animator.SetIKPositionWeight(AvatarIKGoal.RightHand, 1);
            animator.SetIKRotationWeight(AvatarIKGoal.RightHand, 1);
            animator.SetIKPosition(AvatarIKGoal.RightHand, receivedObj.position);
            animator.SetIKRotation(AvatarIKGoal.RightHand, receivedObj.rotation);
```

both functions check whether the lookObj or receivedObj is null or not, before set the weight.


Animator
Controller: Receve
Avatar: tyAvatar
Apply Root Motion: ✓
Update Mode: Normal
Culling Mode: Cull Update Transforms

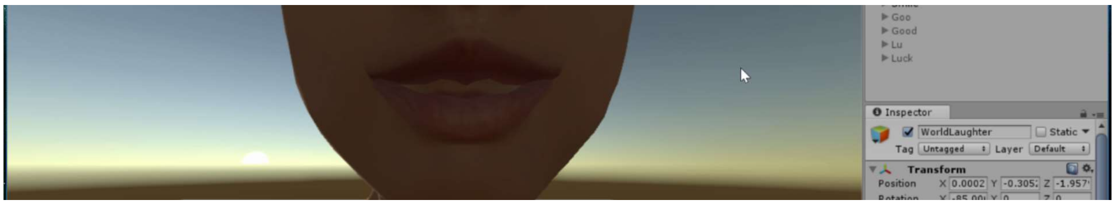
```
Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic:
0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%)
Stream: 0 (0.0%)
```

Receive IK (Script)
Script: ReceiveIK
Ik Active: ✓
Received Obj: TopHat (Transform)
Look Obj: TopHat (Transform)

Fig. 17. Animator with IK enabled and script on the boy

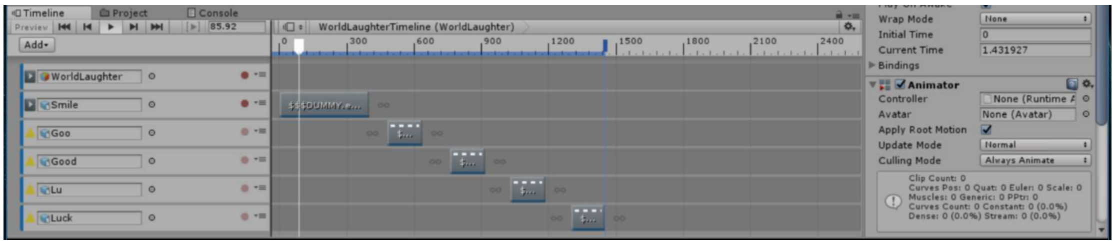
GL2

The only scene in GL2 has been implemented to show the smile of the boy, as well as to make the boy mouth out the title of the story, "Good Luck".



WorldLaughter ☐ Static
Tag: Untagged Layer: Default
Transform
Position X 0.0002 Y -0.305 Z -1.957

Fig. 18. the smile



Wrap Mode: None
Initial Time: 0
Current Time: 1.431927
Bindings

Animator
Controller: None (Runtime A
Avatar: None (Avatar)
Apply Root Motion: ✓
Update Mode: Normal
Culling Mode: Always Animate

```
Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0
Muscles: 0 Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%)
Dense: 0 (0.0%) Stream: 0 (0.0%)
```

Fig. 19. the timelines of the five heads with different shape keys

5 heads of Anakin[1] has been added in the only scene of GL2, all five at the same position, with camera aiming solely at the mouth of the model[2] - because in the original story, there is the sentence "children in the entire world would be laughing", thus, the huge smile shall spread out and fill in the entire screen. All five of the head models have their respective animation keyframes modelled in Blender using Shape Keys function, also, their respective timelines and the animation clips are set in their respective positions on the timeline. Careful modelling has been given while adding shape keys to each expression, for instance, in order

[1] Free model from archive3d.net, https://archive3d.net/?a=download&id=440a1067

[2] In the downloaded model, textures are not properly wrapped, and the eyes are separated from the head. Although only the mouth movement has been captured by the camera, the entire head has been re-grouped up in Blender and applied textures on eyes, face and head.

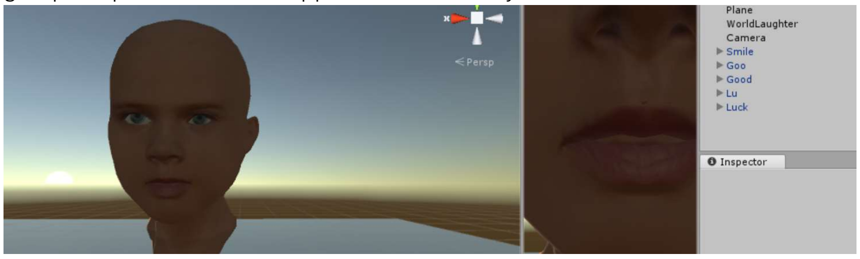


Fig. 20. the full-size Anakin head in Unity

to model a smile on the face, the muscles around the corners of the mouth, philtrum, cheeks, around both corners of the eyes and around the ears have been changed.

Script "Spell.cs" controls which head is currently active.
The first head shows a facial expression from neural, to "surprised", to "happy smile".
The next four heads model shows the facial expression with four different lip-shapes for "Goo-", "-d", "Lu-", "-ck", pressing keys g/d/l/k at the time while the animation clips play, viewers can see the switch from those different lip-shapes.

Animation principles demonstrated in this scene:
Ease in, ease out: the movement of the lips.
Interactive element: the faked lip synchronization according to keyboard events.

Youtube video
GL1: https://youtu.be/ekx8kfXrmiw
GL2: https://youtu.be/BWKiHZLiV14