Falcon City: An Integration of Novint Falcon in Virtual Reality

Jihee Hwang*, Cheng Chen†
Stanford EE 267, Virtual Reality, Course Report, Instructors: Gordon Wetzstein and Robert Konrad

Abstract

Haptic Devices enable various tactile feedback and allow users to interact with a virtual environment in a more immersive way. The current Virtual Reality (VR) interface in the market is largely comprised of devices designed for visual and auditory feedback, with Head Mounted Display (HMD) devices such as Oculus Rift leading the industry by providing significant visual immersion. However, there has been relatively few devices that attempt to utilize a haptic device with a virtual reality setting. In this course project, we integrate a cost-effective haptic device Novint Falcon with an HMD device to improve immersion beyond the visual sense. The VR content is designed to provide a seamless experience with the Falcon, taking into considerations the disadvantages that the Falcon might present in a VR environment.

Keywords: Haptic devices, Novint Falcon, Virtual Reality

1 Introduction

The tactile sense comprises one of the five primary senses of the human body. However, the current methodology of creating haptic feedback from physical input (ex. motors) creates serious constraints in the range of possible movement. Due to this fact, haptic devices are often overlooked in a virtual reality interface compared to visual and auditory senses.

Despite its disadvantages, haptic feedback can add a whole new layer of realism to a virtual environment. Our primary goal is to create a virtual environment that is seamlessly integrated with existing haptic devices on the market.

We have chosen the Novint Falcon as shown in Figure 1 as our primary haptic interface. An affordable price and portable size makes Novint Falcon the most accessible haptic device on the market.

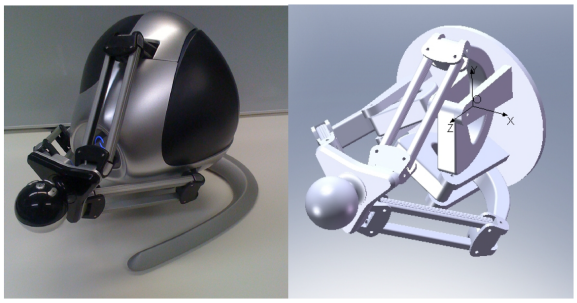

Figure 1: *Picture of Novint Falcon and its CAD file [Martin and Hillier 2009].*

The Novint Falcon [Martin and Hillier 2009] operates with 3 sets of motors and encoders, allowing 3-DOF movement with positional, but not rotational, freedom. The Falcon supports a low-level interface that accepts vectors of force to be inputted each frame, which is then translated internally to force on each of the motors.

*e-mail:jiheeh@stanford.edu
†e-mail:cchen91@stanford.edu

2 Related Work

Figure 2 presents other current state-of-art haptic devices on the market. There are largely two different approaches in recreating haptic sensations: force and vibration feedback. While previously there have been a number of standalone haptic devices demonstrations [Renon et al. 2013; Wei et al. 2014] as well as some attempts to integrate them with a VR environment [Júnior et al. 2012], the latter have largely limited the use of haptics to simple grabbing motions. As such, our purpose in this project is to incorporate the Novint Falcon into an interactive 3D scene with different texture feedback for a more diverse haptic experience.

Figure 2: *Examples of state-of-art haptic devices, which offer texture, force or acoustic feedback.*

3 Approaches

Haptic rendering, as shown in Figure 3 [Salisbury et al. 2004], adds an additional bidirectional path between the human operator and simulation engine (Unity in our case), which forms a feedback loop within the path. Thus, one of the key requirements for haptic rendering is a high frame rate, one that is much higher than what is

required for visual rendering. For example, the Novint Falcon requires in minimum around 120 Frames Per Second (FPS) of haptic feedback, while it can support up to a maximum rate of 1000 FPS; on the contrary, 60 FPS is enough for a smooth visual experience. Therefore, an ideal solution for efficiency would be to separate the audio-visual rendering path and the haptic rendering path. However, we did not find an easy solution for Unity to directly support two separate rendering paths for different devices on the same scene. Therefore, in our project the frame rate for Novint Falcon is equal to that of the audio-visual rendering, determined by the capability of the graphic cards.

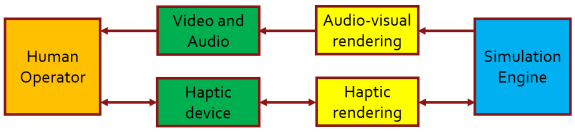

Figure 3: *Illustration of haptic rendering: on the lower path, the bi-directional flow forms a feedback loop between human operator, haptic device and simulation engine. (Adapted from [Salisbury et al. 2004])*

3.1 Interface between Unity and Novint Falcon

One of the major challenges in using a Novint Falcon with state-of-the-art engines and virtual reality devices was the integration of its outdated drivers and SDK. The device was released in 2007, and the company has stopped the support of the device not long after the release. After a long research to find a solution, we were able to integrate the Unity Engine with the Falcon's default SDK, which provided the following low-level functions:

- Getting the position of the device and whether a button is pushed down;
- Setting next device position `SetServoPos();`
- Setting next device force `SetServo()`.

However, the SDK did not include any higher-level functionalities in simulating different haptic sensations. While there exists a few open-source haptic libraries such as chai3D, it was difficult to utilize it with the Unity Engine as well. Thus, our further challenge was to develop our own set of methods for that purpose.

3.2 Texture feedback creation

We created four features to demonstrate the integration of Novint Falcon with Unity, as shown in Figure 4. Solid surface represents a simple blocking object. Water and sand allows the user to move through the area but with added sensations that simulate each of the respective elements. Pop recreates a sense of plucking, similar to the sense of removing a magnetic object from a strong magnet.

In Unity, we assigned the Unity class `Collider` to a hand model which corresponds to the end-effector of the haptic device. We have also created three diffrent colliders for surface, water and sand, using Unity's tag feature to assign each one to different objects on the scene. When an object touches, enters or exists a collider, the Unity class `Collider` calls functions `OnCollisionEnter()`, `OnCollisionStay()`, and `OnCollisionExit()`. Under different functions, we read the physical position of the Novint falcon from its server and the object the user is interacting with, and then set the control method of the Novint Falcon, forming a feedback loop. The script that calls each of the different effects is shown below.

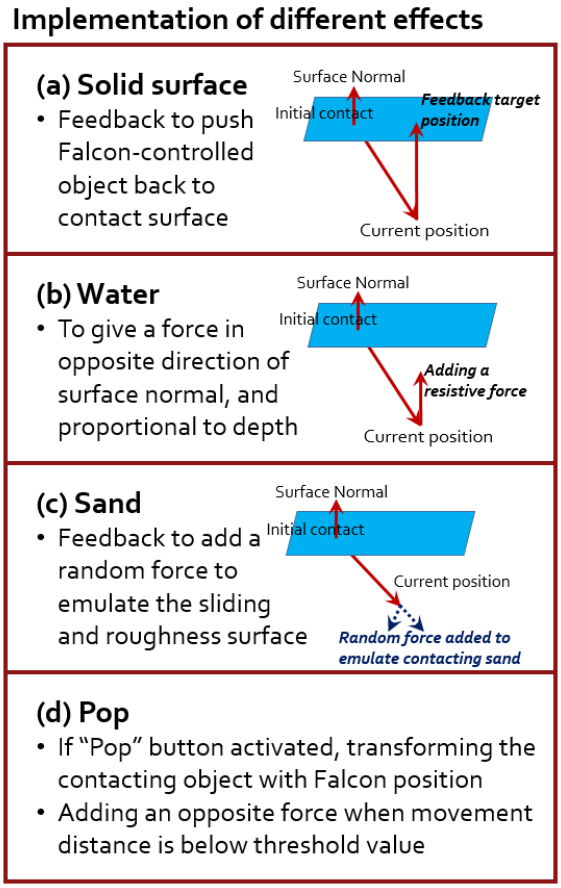

Figure 4: *Effects implemented for Falcon device: (a) position feedback when touching a solid surface, guiding the user's position along the surface; (b) force feedback when touching water, adding a resistive force in the opposite direction to simulate viscosity; (c) force feedback when touching sand, adding a minimal force in a random direction to emulate roughness from grains; (d) mechanism of implementing grabbing effects using the button on Falcon device.*

```
private void OnCollisionStay(Collision c)
{
  // Position processing
  ..
  if (c.collider.tag == "Surface")
  {
    isBuildingInTouch = true;
    buildingInTouch = c.collider.gameObject;
    surfaceHaptics(c, vectorProj);
  }
  else if (c.collider.tag == "Water")
    waterHaptics(c, vectorProj);
  else if (c.collider.tag == "Sand")
    sandHaptics(c, vectorProj);
  else if (c.collider.tag == "PopEffect")
    popHaptics(c, vectorProj);
}
```

The functions that implements different tactile effects are shown below.

```
void surfaceHaptics(Collision c,
                    Vector3 vectorProj)
{
```

```
  Vector3 normalOutsidePos =
           c.contacts[0].point - vectorProj;
  PosX = normalOutsidePos.x;
  PosY = normalOutsidePos.y;
  PosZ = normalOutsidePos.z;
  Strength = surfaceStrength;
}
void waterHaptics(Collision c, Vector3
                                  vectorProj)
{
  SpeedX = -viscocity * vectorProj.x;
  SpeedY = -viscocity * vectorProj.y;
  SpeedZ = -viscocity * vectorProj.z;
}
void sandHaptics(Collision c, Vector3
                                  vectorProj)
{
  Vector3 normalOutsidePos =
  (c.contacts[0].point-vectorProj).normalized;
    float posrand = Random.value * 0.01f;
    PosX = normalOutsidePos.x + posrand;
    PosY = normalOutsidePos.y + posrand;
    PosZ = normalOutsidePos.z + posrand;
    Strength = surfaceStrength;
}
void popHaptics(Collision c, Vector3
                                  vectorProj)
{
  Vector3 vectorMoved = GetServoPos() -
                origContactObjectPoint;
  float distanceMoved = vectorMoved.magnitude;
  if (distanceMoved < 0.5)
  {
    SpeedX = contactNormal.x*distanceMoved*15;
    SpeedY = contactNormal.y*distanceMoved*15;
    SpeedZ = contactNormal.z*distanceMoved*15;
    c.collider.transform.position =
                                GetServoPos();
  }
  else if (distanceMoved < 1)
  {
    SpeedX = 0;
    SpeedY = 0;
    SpeedZ = 0;
    c.collider.transform.position =
                                GetServoPos();
  }
}
```

3.3 Scene creation

A demonstration that considers the limited spatial freedom of the Falcon device was required to create a more realistic VR environment. As such, we were inspired by a SimCity-like top-down interface, which naturally leads the user to perform small, yet precise, movements. In addition, a setting that is a whole "city" can allow various textures and natural elements to be incorporated in the scene. Figure 5 captures the main components in the complete scene we created for the scene.

We also added a skybox of outer space to add realism considering the context of the scene, as shown in Figure 5(a). In the main scene Figure 5(b), we have placed sea, terrain and buildings for water, sand, and solid surface textures, respectively. When touching different bodies of textures, the feedback from Novint Falcon varies. In addition, being able to pluck the buildings from a ground and start an apocalypse adds a good degree of interactivity and entertainment to the demo as well.

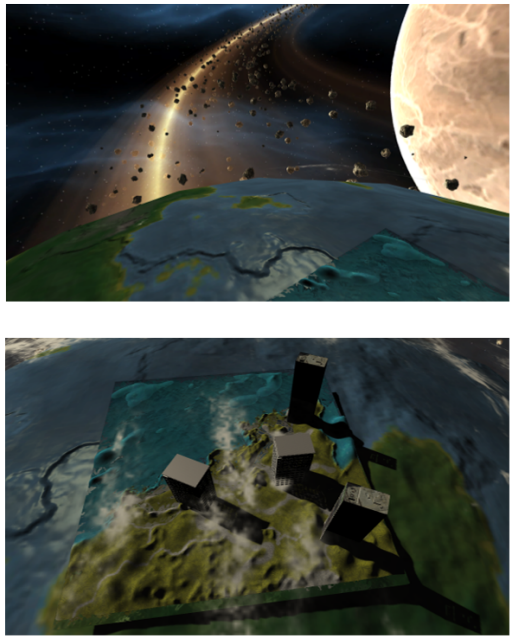

Figure 5: *Created scene in unity: (a) background and surrounding of the experience; (b) main scene with different textures, including sand, solid surface (building top) and water.*

4 Results

We first evaluate the standalone scene in Unity, and then the integration between the Falcon and the Unity engine.

The scene starts out with a frame rate of around 110 FPS, which provides reasonable, but not optimal, haptic experience. However, as the "apocalpyse" event begins, multiples of fire prefab objects and various particle effects are added in the scene, dropping the frame rate significantly to around 30FPS. Such rate renders the haptic experience nearly unusable.

After integrating Novint Falcon into the scene, the HMD is no longer able to track the head movement due to the incompatibility with the IMU. Therefore, the final result is an integration of Novint Falcon with a stationary 3D scene.

We do manage to get distinct force feedback when touching different surfaces. The overall accuracy of the feedback is good when the scene is not complicated and its frame rate of is high. However, once the fire effect is turned on, a lot of irregular vibration of Novint Falcon is observed.

5 Analysis and Discussion

Overall speaking, most of the feedback we obtained during the demonstration session are positive expressing the interesting experience adding more interaction with the scene. However, there are several limitations of the current setup, including incompatibility issues, the mechanic properties of Novint Falcon, the interface between the device and Unity, and the haptic effects function we wrote.

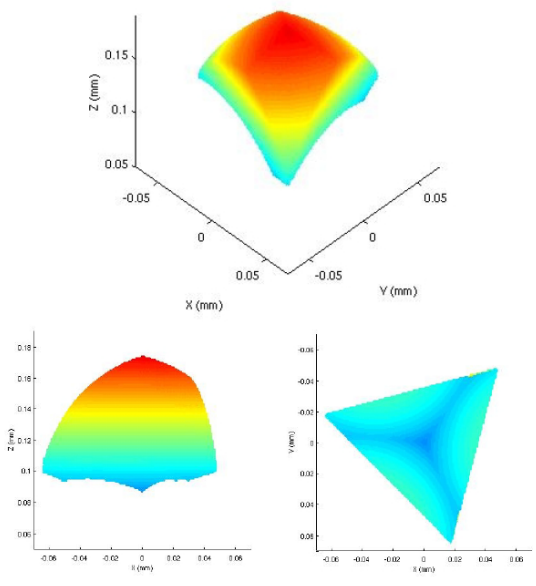

Figure 6: *Work space range of Novint Falcon in x, y and z direction [Martin and Hillier 2009].*

5.1 Incompatibility issues

There are basically several incompatibility issues that preventing us to create a fully functional virtual reality environment with Novint Falcon.

- Novint Falcon driver only supports 32-bit environment: the control library file we obtained can only support 32-bit environment. Moreover, the library in Novint Falcon SDK is in 32-bit, we tried in vain to compile the dll file into 64-bit environment version.
- Novint Falcon's conflict with IMU; when two devices are opened in the same Unity project, the lagging makes it unusable. While the reason is not clear, we suspect that it may be due to the conflict of different versions of FTDI driver. Both Novint Falcon and arduino use FTDI drivers, but the former use a version dating back almost 10 years whereas the latter uses the newest version.
- Oculus Rift's conflict with 32-bit Unity: we opened the same project with both 64-bit and 32-bit unity. The 64-bit unity works with Oculus well, whereas 32-bit Unity cannot recognize the device at all.

Due to these incompatibilities, we ended up demonstrated a 3D scene with Novint Falcon. The HMD only helps create the 3D experience without the capability to track head motions.

5.2 Mechanic properties of Novint Falcon

One major limitation of Novint Falcon is its limited work space as shown in Figure 6 [Martin and Hillier 2009]. As a result, the movement in the scene is limited a very small range.

5.3 Interface

In current interface implementation, Unity directly sends the control signal to Novint Falcon. Therefore, they share the same frame rate. However, for Novint Falcon to give force and position feedback well, the refresh rate should be more than 120FPS, which is demanding for our scene, especially when more objects are added for special feature.

In our scene, at the beginning, the frame rate is about 110 FPS, and the device behave mostly as it is. Once the fire effects are initiated, the frame rate drops to about 30 FPS, and the device has a lot of unexpected vibrations.

5.4 Haptic effects functions

In the project, we create our own haptic effects script. Due to the limitation of accessibility of Novint Falcon functions, we only manage to create some rudimentary effects with non-negligible simplification from physics. Such simplification also generates certain aberrations even when other factors are ideal.

6 Conclusion

During the demonstration session, we got mainly good feedback expressing the interesting experience of a more interactive experience. This definitely shows the benefits of integrating haptic devices with VR environment. We believe that, with a more up-to-date haptic devices and better haptic effects library, a much better interactive and realistic VR environment is possible.

Acknowledgements

We thank Professor Wetzstein and Robert Konrad, for their instructions and help during the course.

References

JÚNIOR, A. R. S., CARDOSO, A., JÚNIOR, E. A. L., AND FARO, A. D. M. 2012. The Use of Virtual Reality for Simulation and Training of Bovine Artificial Insemination with Haptic Devices. In *Symposium on Virtual and Augmented Reality (SVR)*, 66–73.

MARTIN, S., AND HILLIER, N. 2009. Characterisation of the Novint Falcon Haptic Device for Application as a Robot Manipulator. In *Australasian Conference on Robotics and Automation (ACRA)*, 1–9.

RENON, P., YANG, C., MA, H., AND CUI, R. 2013. Haptic Interaction Between Human and Virtual iCub Robot Using Novint Falcon with CHAI3D and MATLAB. In *Chinese Control Conference*, 6045–6050.

SALISBURY, K., CONTI, F., AND BARBAGLI, F. 2004. Haptic Rendering: Introductory Concepts. *IEEE Computer Graphics and Applications 24*, 2, 24–32.

WEI, L., NAJDOVSKI, Z., ZHOU, H., DESHPANDE, S., AND NAHAVANDI, S. 2014. Extending Support to Customised Multi-Point Haptic Devices in CHAI3D. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 1864–1867.