MRI Simulator Project

https://sites.google.com/site/mrisimulator/

http://youtu.be/LFvszmBZJZQ

Klara Proffen
Jamie Lindsley

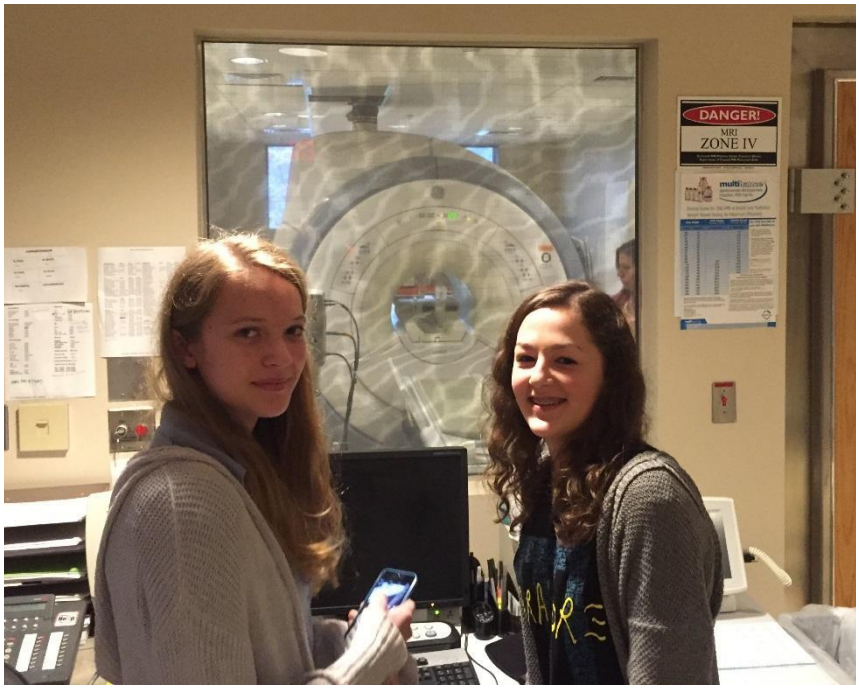
Jefferson Middle School, Oak Ridge, Tennessee

Introduction

In the book Divergent by Veronica Roth [1] in one scene one of the main characters Four has to face his fears as part of his initiation for Dauntless. One of his fears is claustrophobia and both of us do not like tight spaces. Claustrophobia is the fear of crowded spaces. People suffering from claustrophobia won't go into elevators/lifts, use changing rooms, tunnels, basements/cellars, subway trains (UK: tube/underground trains), small rooms, hotel rooms with windows that do not open, revolving doors, airplanes, public toilets, locked rooms, cars - especially if they have central locking, trains, crowded areas, automatic car-washes, and MRI scanners.

This gave us the idea for this project related to global health. Magnetic resonance imaging (MRI) is done for many reasons. It is used to find problems such as tumors, bleeding, injury, blood vessel diseases, or infection. MRI also may be done to provide more information about a problem seen on an X-ray, ultrasound scan, or CT scan. MRI scans are very important and we wanted to find a way to help people overcome their fears.

We wanted to learn more about MRI scanners and patient fear and had the opportunity to visit the Methodist Medical Center and see a MRI. We learned that MRI scanners cost between one and two million dollars and that you cannot have any metal in the room or it gets sucked into the scanner. This makes it hard to use a real scanner to just try how it feels. The nurse told us that especially larger people are scared of the tight space they get squeezed into.

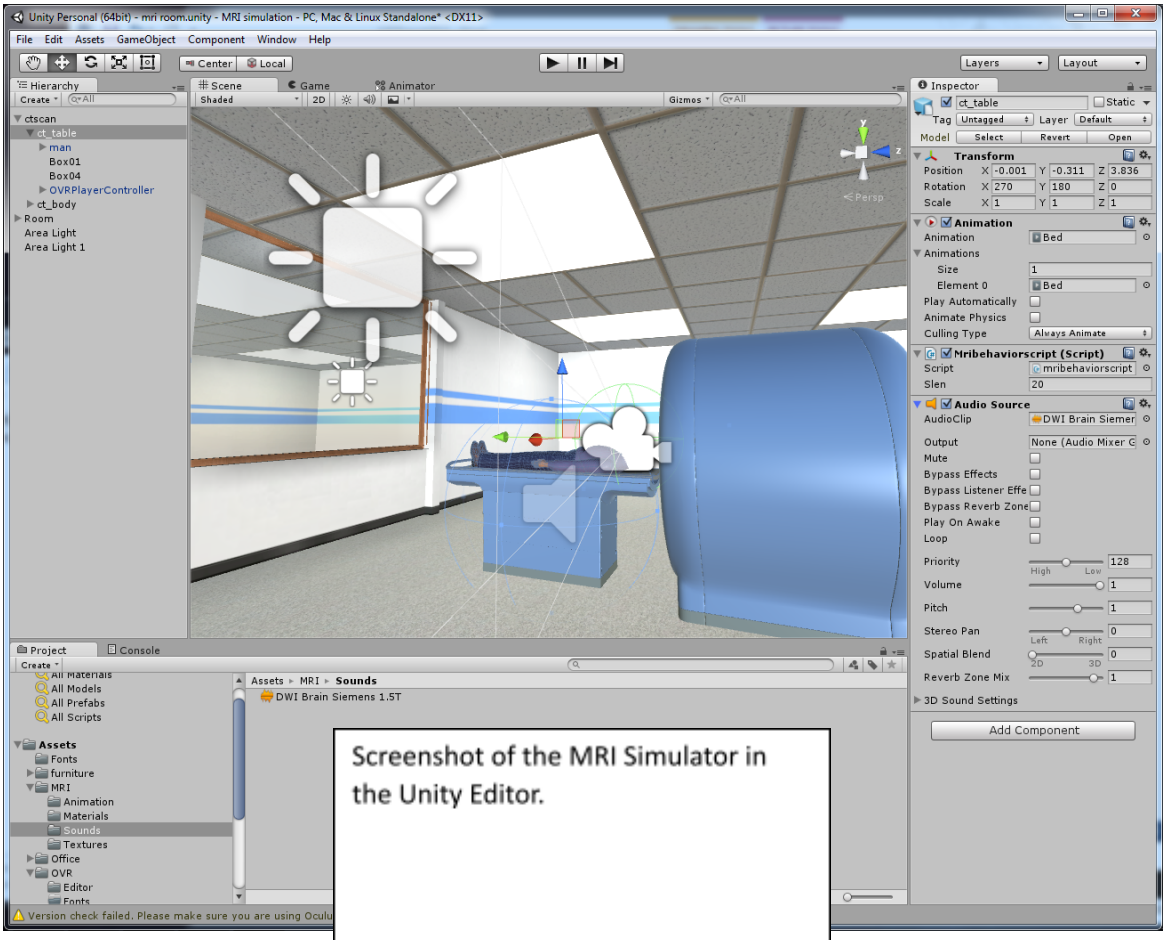


Project team Klara and Jamie learning about MRI scanners at the Methodist Medical Center in Oak Ridge, TN.

Technology summary

We wanted to help claustrophobic people to overcome their fear of being in tight spaces when being inside a MRI scanner. Recently we could play with a virtual reality headset, the Oculus Rift. We played awesome games such as Titans of Space [2]. This gave us the idea to create a virtual reality MRI simulator that feels real enough to help people overcome their fear of being inside a MRI scanner.

To make video games and virtual reality simulations we used the program Unity [3] which is free for personal use. Using Unity is much easier than we thought, although we had some issues at first with lights and navigation in the 3D scenes.



Screenshot of the MRI Simulator in the Unity Editor.

The 3D models used in our si number of sites [4]. Some of the models such as the picture frames we modified and put our own images on and we made our own medical certificates to hang on the wall.

The Oculus Rift [5] is awesome! The Rift has a phone screen on the inside and two lenses, one for each eye. The software creates an image on the screen for each eye from two cameras separated by the eye distance. This creates the three dimensional appearance. Inside the Rift

headset there are gyro sensors just like in our iPhones. These are used to figure out how your head is moving and update which part of the simulation is seen. This makes it feel like you are really inside the virtual world - if you do not believe it try Don't Let Go [6].

Did we need to program? Yes! After putting all the 3D models into the scene in Unity, we needed to program the behavior of the scanner: Moving the bed into the scanner, playing the sound of the running scanner which we downloaded and finally moving the bed back out. We learned the programming language c sharp which is used in Unity to define object behaviors. One difference to other programs we have written (e.g. programming a Lego robot) is that it seems commands are not waiting on each other and we learned about update cycles and the use of coRoutines.

Some parts of the MRI scanner code is shown below and the complete code is shown in the appendix:

```
void Update () {
	if (scanning) {
		if (Input.GetKeyDown (KeyCode.E)) {
			StartCoroutine (EndScan());
		}
		if ((Time.time-time)>slen){
			StartCoroutine (EndScan());
		}
		CheckMove ();
	} else {
		if (Input.GetKeyDown (KeyCode.S)) {
			StartCoroutine (StartScan ());
		}
		if (Input.GetKeyDown (KeyCode.UpArrow)) {
			slen=slen+10.0f;
			monitor.text = string.Concat ("Time\n", slen.ToString(),"s");
			}
		if (Input.GetKeyDown (KeyCode.DownArrow) && slen>10.0f) {
			slen=slen-10.0f;
			monitor.text = string.Concat ("Time\n", slen.ToString(),"s");
		}
	}
}
```

The routine *update* above is called every frame that is sent to the Oculus or 75 times per second. First we check is a scan is running indicated by the Boolean variable *scanning*. If a scan is running, three conditions in the *if* statements will end the scan: (1) the key 'e' is pressed; (2) the scan time has been reached and (3) the patient moved too much which is checked in the routine *CheckMove()*.

If we are not scanning (else branch), we check if the key 's' is pressed and the routine *Scan()* is called using *StartCoroutine*. This will create a separate sub-program that executes the scan. The keys up arrow and down arrow will increase or decrease the scanning time by ten seconds and update the text on the virtual monitor. The *StartScan()* and *EndScan()* routines are shown below:

```
// Starting scan
IEnumerator StartScan(){
	anim [anim.clip.name].time = 0f;
	anim [anim.clip.name].speed = 1f;
	anim.Play ();
	yield return new WaitForSeconds (alen);

	GetComponent<AudioSource> ().Play ();

	time = Time.time;
	forward = oculus.GetComponent<Transform> ().forward;
	scanning = true;
}
```

In the *Scan* routine, the first three lines, set the starting time and speed of the animation that moves the bed into the scanner and play the animation. After waiting for the time the animation takes, the scanner sound is played. Next we store the three head rotation angles from the Oculus, the starting time in variables and finally set scanning to true as our scan has started.

```
// Ending scan
IEnumerator EndScan(){
	scanning = false;
	GetComponent<AudioSource>().Stop();

	anim [anim.clip.name].time = alen;
	anim [anim.clip.name].speed = -1f;
	anim.Play ();
	yield return new WaitForSeconds (alen);
	warning.SetActive(false);

}
```

To end a scan, we set scanning to *false* and stop the sound. Next the bed is moved back out by setting the animation time to the end of the clip (when the bed in) and play the clip backwards by setting the speed to negative one and finally play the clip. We wait until the animation is done and turn the warning message back off in case the scan was aborted.

```
void CheckMove (){
	float move = Vector3.Angle(forward,oculus.GetComponent<Transform>().forward);
	if (move > wiggle){
		warning.SetActive(true);
		StartCoroutine (EndScan());
	}
}
```

The routine *CheckMove* will first calculate the angle between two vectors; the forward direction of the rift when the scan started and the current forward direction. If the angle is larger than the value in wiggle, the warning is displayed and the scan is ended.

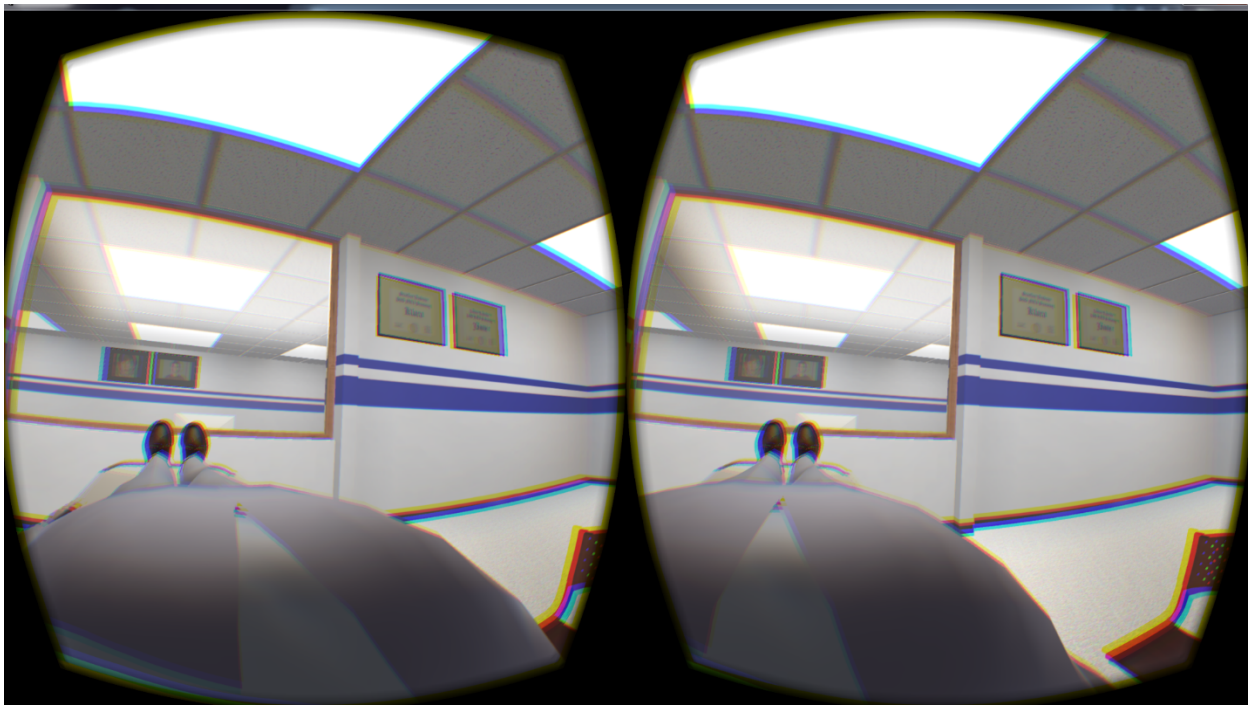
Finally we build the application for distribution and created our MRI simulator website https://sites.google.com/site/mrisimulator/. The program runs on Windows and required the Oculus runtime library to be installed. There is also a version for a normal screen.

Results

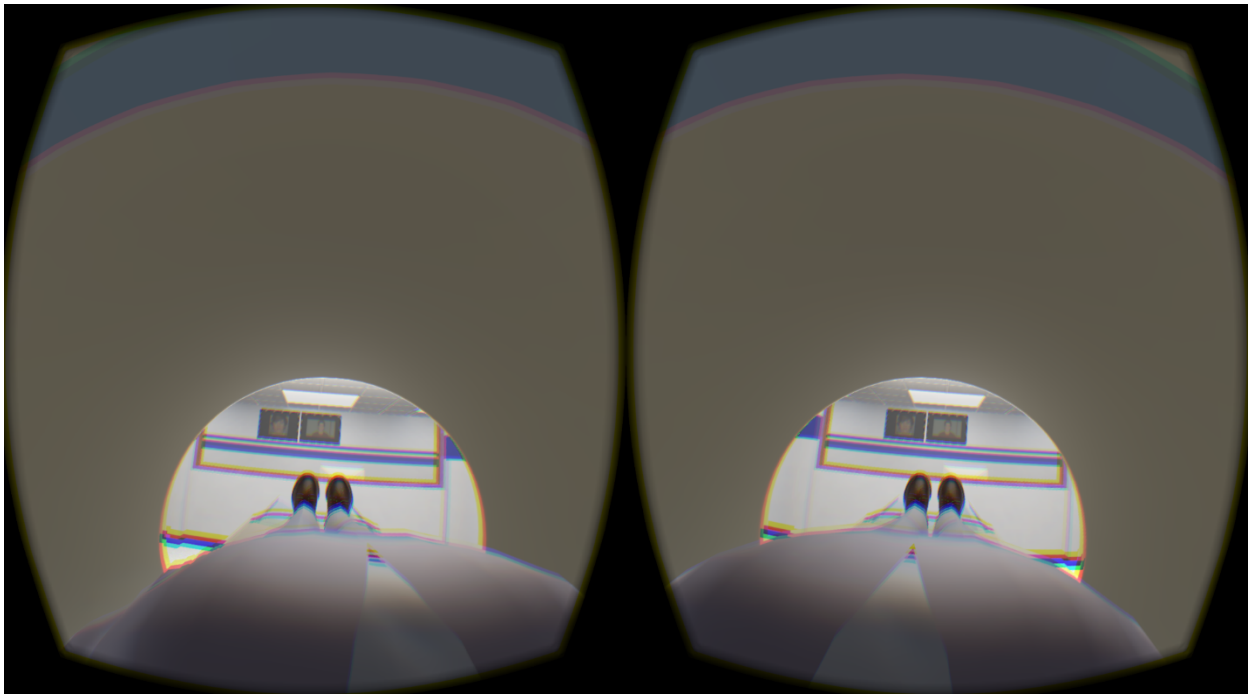
After finishing the program, we both tried out the MRI simulator. It felt like really being there and when the bed entered the scanner, we both felt uneasy because of the tight space. Before it really worked, we had to adjust the position of the camera in the simulation several times, until we had the right view.
Next we need to field test our simulator. We volunteered some of our friends to lie down on a mattress and put on the Oculus Rift and enter the MRI scanner simulation. The Director of Medical Imaging at the Methodist Medical Center invited to return and show them the finished product in a few weeks.

Below is a screenshot of the simulator running. It shows the images for the right and left eye. However, the screenshot does not look as cool and realistic as the experience using the Oculus Rift.

Jamie getting ready to take the virtual MRI simulator scan.

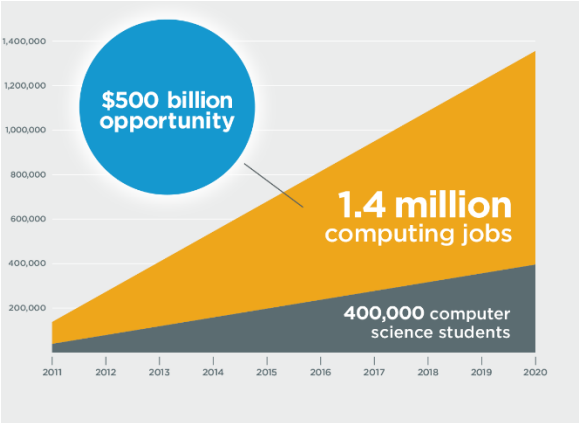


Before going into the MRI scanner.



Inside the MRI scanner.

Why girls in tech



More girls should study computer science for three reasons: first by 2020 there will be one million more tech jobs than students [7]. Second just 12 percent of computer science degrees are awarded to women. Third girls are often more creative and in many tech jobs you need creativity. Our project is a good example, we needed to program the behavior of the MRI scanner, but also create a realistic environment. Did you notice our diplomas on the wall and the pictures of Dylan O'Brien and Hayes Grier? We interviewed a computer scientist

from the Oak Ridge National Laboratory and he told us that when he first started there were more women working with him than now that he has been advancing in his career.

How do we get more girls in computer science? The ProjectCSGirls competition gave us a great opportunity to learn more about computer science, programing, and virtual reality. This made us realize how many exciting career choices there will be for us.

References

[1] Veronica Roth (2011), Divergent,
[2] http://www.titansofspacevr.com/
[3] http://www.unity3d.com/
[4] http://www.turbosquid.com/ , http://archive3d.net/
[5] http://www.oculus.com
[6] https://share.oculus.com/app/dont-let-go
[7] http://www.code.org

Appendix

Complete C Sharp code controlling the behavior of the virtual MRI scanner.

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class mribehaviorscript : MonoBehaviour {

	public float slen = 10.0f;
	public float wiggle = 5.0f;
	public Text monitor;

	private Animation anim;
	private GameObject oculus;
	private GameObject warning;
	private float alen;
	private float time;
	private bool scanning;
	private Vector3 forward;

	// Initialization of values at start
	void Start () {
		anim = GetComponent<Animation> ();
		alen = anim [anim.clip.name].length;
		warning = GameObject.Find ("Warning");
		oculus = GameObject.Find ("CenterEyeAnchor");
		warning.SetActive(false);
		scanning = false;
		monitor.text = string.Concat ("Time\n", slen.ToString(),"s");
	}

	// Update is called once per frame
	void Update () {
		if (scanning) {
			if (Input.GetKeyDown (KeyCode.E)) {
				StartCoroutine (EndScan());
			}
			if ((Time.time-time)>slen){
				StartCoroutine (EndScan());
			}
			CheckMove ();
		} else {
			if (Input.GetKeyDown (KeyCode.S)) {
				StartCoroutine (StartScan ());
			}
			if (Input.GetKeyDown (KeyCode.UpArrow)) {
				slen=slen+10.0f;
				monitor.text = string.Concat ("Time\n", slen.ToString(),"s");
			}
			if (Input.GetKeyDown (KeyCode.DownArrow) && slen>10.0f) {
				slen=slen-10.0f;
				monitor.text = string.Concat ("Time\n", slen.ToString(),"s");
			}
		}
	}
```

```
        // Check for head movement
        void CheckMove (){
                float move = Vector3.Angle(forward,oculus.GetComponent<Transform>().forward);
                if (move > wiggle){
                        warning.SetActive(true);
                        StartCoroutine (EndScan());
                }
        }

        // Starting scan
        IEnumerator StartScan(){
                anim [anim.clip.name].time = 0f;
                anim [anim.clip.name].speed = 1f;
                anim.Play ();
                yield return new WaitForSeconds (alen+0.5f);

                GetComponent<AudioSource> ().Play ();

                time = Time.time;
                forward = oculus.GetComponent<Transform> ().forward;
                scanning = true;
        }

        // Ending scan
        IEnumerator EndScan(){
                scanning = false;
                GetComponent<AudioSource>().Stop();

                anim [anim.clip.name].time = alen;
                anim [anim.clip.name].speed = -1f;
                anim.Play ();
                yield return new WaitForSeconds (alen+0.5f);
                warning.SetActive(false);

        }
}
```