Aperture Bioscience Laboratories

Progress Report 7 for NICView: A Virtual NICU Simulation

Due: 2/27/15

Professor Walker

Amy Young, Caitlin Li, Jennifer Duan, Pamela Wu, Lindsey Sumners

I. Background of the project

Considering the new regulations that limit work hours for medical residents, these residents do not have the same real case experience as in the past. Therefore, this reduces patient safety, because the residents do not experience a large and varied amount of cases. In order to solve this problem, our project will involve developing a simulation game that can be played at the resident's home. By playing this game, the residents will be able to gain better conceptual knowledge by going through different scenarios, which addresses the issues of volume and variety of cases and patient safety. By having this knowledge, the residents will be able to use the time in the hospital in order to gain experience with the physical actions in medicine.

II. Achievements since the last report

The two main areas of progress since the last report has been in graphics and programming. In the area of graphics, three different pieces of medical equipment have been drawn: a breathing tube, red blood cell bag, and a towel. Also, the vitals monitor has been fully colored. All of these can be seen in Figures 1 and 2.

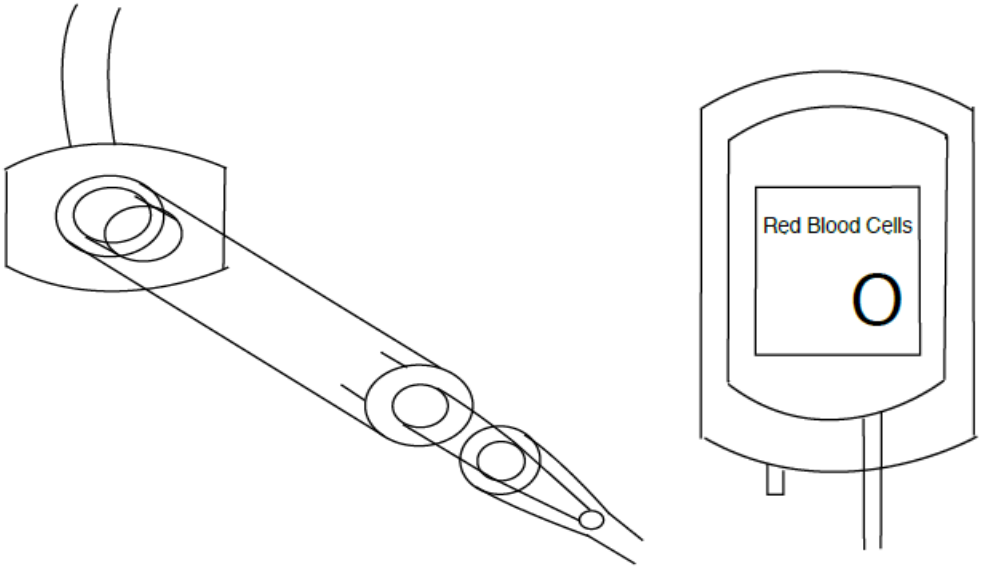

Figure 1: A) Breathing Tube B) Red Blood Cell Bag

Figure 2: A) Towel B) Colored Vitals Monitor

The other area of progress has occurred in programming. All of the code that is being used in the project is being shared through the use of CodeShare and can be seen at the following link: http://www.codeshare.io/ybJtV. The first part of programming has been the creation of a timer, so the time that the residents take to make a decision can be monitored. All of the code that is related to the timer is shown below.

```
// timer code-couting. NOT Timer game object
using Unity.Enginer;
using System.Collections;

// create a variable that initiates time
public var startTime; var textTime : String
 void Update () {
 if isTiming{
 startTime = Time.time;
 }
}

// Create a minute, seconds, and percentage of total time that counts up from zero
function onGUI () {
 var guiTime=Time.time-startTime;
 var minutes : int = guiTime / 60; var seconds : int = guiTime % 60; var fraction : int = (guiTime
* 100) % 100;
   text = String.Format ("{0:00}:{1:00}:{2:000}", minutes, seconds, fraction);
 GUI.Label (Rect (400, 25, 100, 30), textTime);

// Timer begins when the "PROCEED" button is clicked:
//Tests for whether or not the "Proceed" button has been clicked
                    if (GUI.Button (buttonSize, buttonText)) {
```

```
                    //Destroys the button (this) if it has been clicked
                    Destroy (this);
                    //Sets isTiming to True to initialize timing in Update function
                    isTiming = true;

//Code for resetting the timer upon appropriate collision:
void OnTriggerEnter(Collider btube) {
            //Test for collision with a gameObject that is tagged "baby"
            if (btube.gameObject.tag == "baby") {
                    //If collision is detected
                    Debug.Log ("Collision between breathing tube and baby detected");
                    print ("Good job. The correct first step has been taken and the meconium
has been suctioned out. Please proceed with the rest of the scenario.");
        startTime = Time.time;
```

Also, the medical inventory has been started. Two different pieces of code are related to this. The first piece of code is the naming of different objects within the game. The second piece of code is the beginning of making the list that will comprise the medical inventory.

```
// creating the variables identifying item descriptions, properties, and categories
using UnityEngine;
using System.Collections;

[System.Serializable]
public class Item {
        public string itemName;
        public int itemID;
        public string itemDesc;
        public Texture2D itemIcon;
        public ItemType itemType;

        public enum ItemType {
                Airway,
                IV,
                Medication,
                Other
        }
}

// creating a list
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class ItemDatabase : MonoBehaviour {
        public List<Item> items = new List<Item>();
```

```
}


//Code for resetting the timer upon appropriate collision:
void OnTriggerEnter(Collider btube) {
                //Test for collision with a gameObject that is tagged "baby"
                if (btube.gameObject.tag == "baby") {
                        //If collision is detected
                        Debug.Log ("Collision between breathing tube and baby detected");
                        print ("Good job. The correct first step has been taken and the meconium
has been suctioned out. Please proceed with the rest of the scenario.");
        startTime = Time.time;
                }

Finally, the last piece of code to be discussed is the sample code that shows how the points will
be updated and displayed during the game. It also shows how the points are set to zero at the
beginning of a scenario.

//Initial points code:

  public GUIText scoreText;
   private int score;
  //AddScore function with input int: newScoreValue
  public void AddScore (int newScoreValue)
   {
     //Add newScoreValue to score
         score += newScoreValue;
         //Run UpdateScore function
      UpdateScore ();
   }

   void UpdateScore ()
   {
    //Update GUI score display
    scoreText.text = "Score: " + score;
   }
}
void Start ()
   {
      //Initialize score as 0 at startup
                score = 0;
                //Run UpdateScore function to display score
      UpdateScore ();
   }
```

III. Deviation from the plan and corrective action

Looking at the work plan, the team is still a little bit behind in the fact that the first two scenarios are not completely coded. However, since the addition of two more people on programming, the progress in this area has improved greatly. Therefore, the team should be back on track within the next couple of weeks. Also, during the week, it was realized that Pam was not given all of the needed medical equipment. Therefore, each person who completed a flowchart sent Pam a list of needed equipment. Due to this, there should be no more problems in graphics that will impede its progress.

IV. Plan for Next Week

For next week, the team plans to have a few main tasks completed. The first is that the team wants to finish drawing the needed medical equipment. This task is important, because all of these items are needed to finish the inventory. In the area of programming, the medical inventory should be completed in order that the player will be able to choose pieces of equipment from it. Also, the code for assigning points to different actions will be completed. This will allow the team to work on integrating the point system and timer together, which will allow the player to receive a time-weighted score. The team will also be testing sequential collisions to make sure that points are given in the correct order. Finally, code will be started that will display the relevant information to the vitals monitor.

V. Assessment of Progress

Considering the current progress, the team is almost back on track to complete the various tasks on time. If the team continues to have the increased progress in programming like this week, the project will be able to be completed on time.