

Livrable G – Prototype II et rétroaction du client (FA-01)

Jacob Godin, Benoît Gratton, Jacob D'Aoust, Huber Lachaine et Christophe Lepage Bilodeau

Le jeudi 14 novembre 2019

Introduction	3
Pourquoi est-ce qu'on fait cet essai?	3
Description des objectifs de l'essai	3
Qu'est-ce qu'on va faire et comment?	3
Le prototype	4
Points d'amélioration pour les prototypes (1 et 2)	4
Conclusion	4

Introduction

Ce livrable est constitué de la planification, la construction et l'essai du deuxième prototype du simulateur de tracteur. La rétroaction de ce prototype présenté dans le document présent sera cruciale afin d'optimiser la performance du dernier prototype. Dans ce livrable, nous allons décrire la raison du prototype, les objectifs d'essais, une description du prototype, comment concevoir le prototype et l'amélioration du prototype.

Pourquoi est-ce qu'on fait cet essai?

Cet essai a pour but de concrétiser le logiciel informatique que nous utiliserons pour le simulateur. Avec le prototype 2, nous pourrions déterminer les points forts et les points à améliorer du logiciel informatique utilisé. Cela correspond à notre deuxième sous-système: l'interface utilisateur. De plus, avec le système Arduino, soit les boutons et manettes commandés en ligne, le prototype nous donnera une bonne indication du fonctionnement des fonctions interactives du simulateur.

Description des objectifs de l'essai

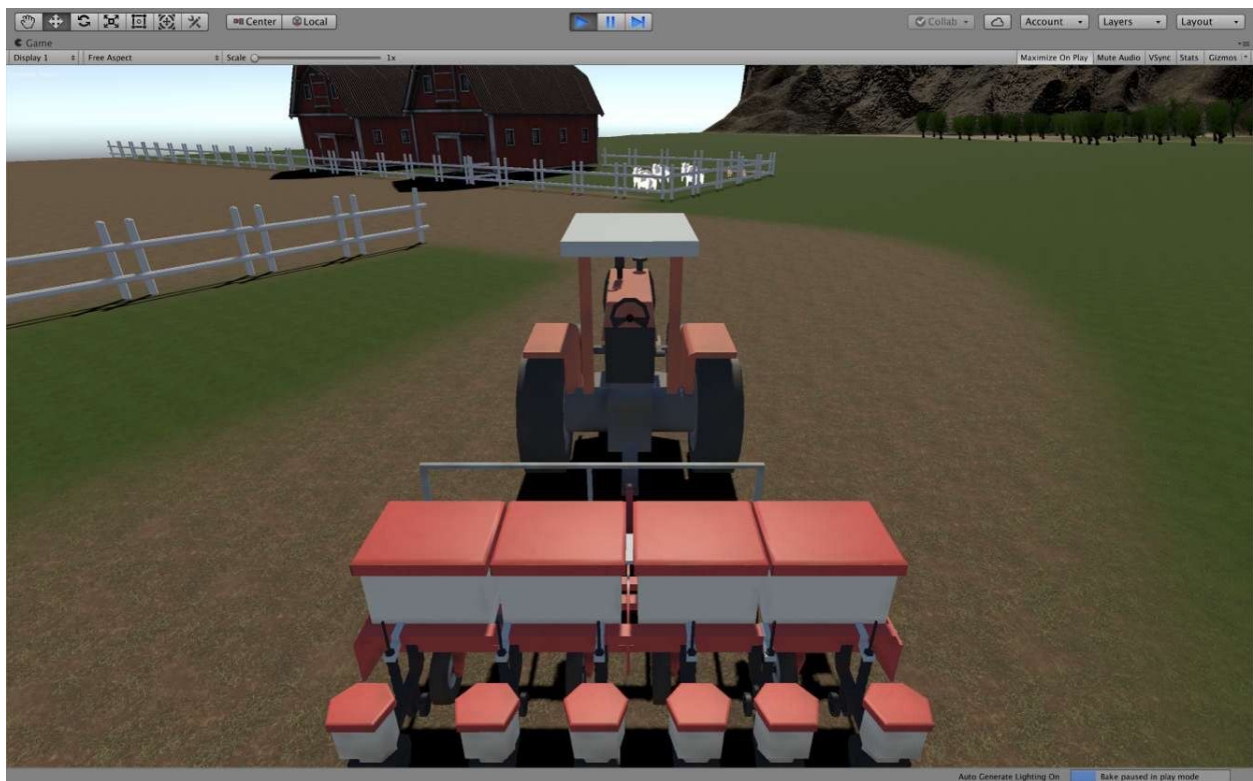
L'essai a pour but de vérifier la faisabilité du logiciel impliquée pour la simulation, ainsi que l'implémentation des mécanismes comme les boutons, à la simulation. L'objectif principal est d'assurer que le programme de simulation est assez accessible pour tous les utilisateurs et qu'il respecte bien les critères de conception définis (tel que la durée, la difficulté, l'aspect éducatif, etc.). Les essais de notre prototype cesseront seulement lorsque tous ces critères soient atteints.

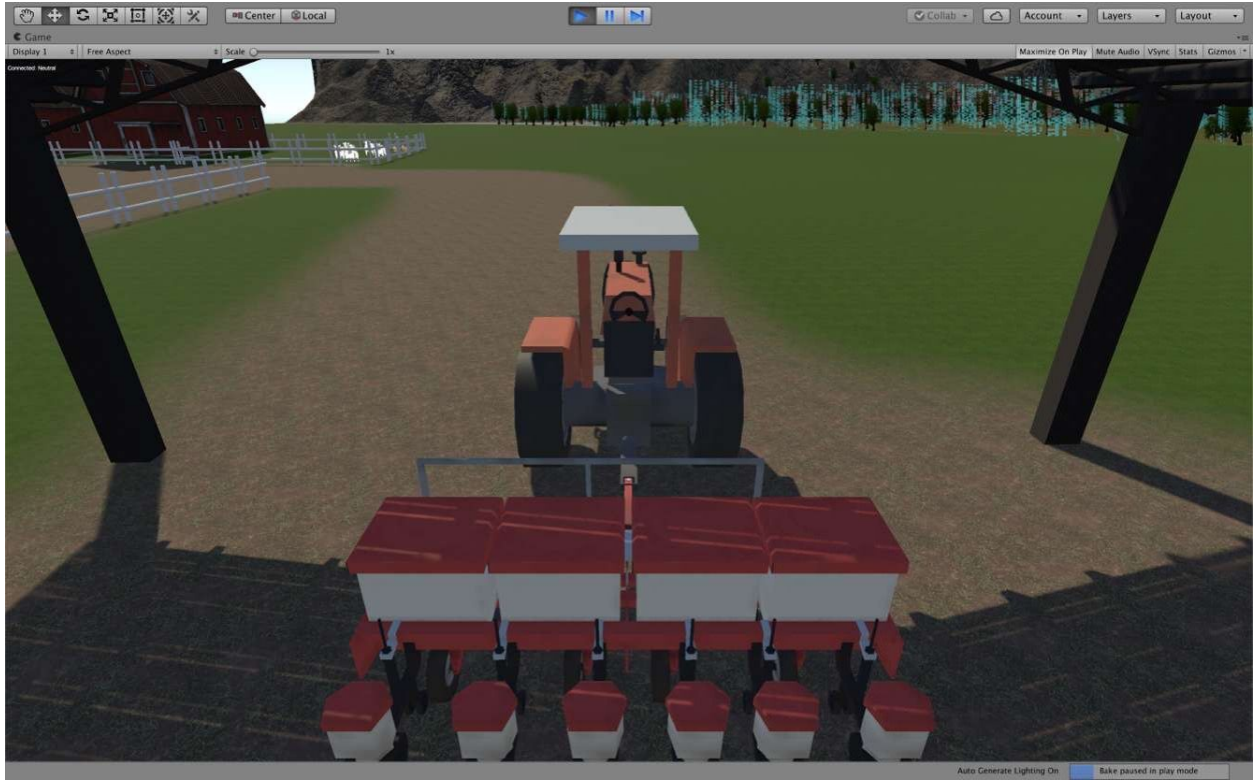
Qu'est-ce qu'on va faire et comment?

Suite à la création de la simulation et l'implémentation des boutons et du volant, nous allons tester le programme et vérifier si la simulation fonctionne comme désirée, c'est à dire que le volant fasse tourner le tracteur de la simulation, que les boutons aient les fonctionnalités désirés et que la simulation fonctionne bien. De plus, l'essai sera un succès si la simulation est courte, amusante, et interactive, tout à la demande du client.

Le prototype

Ce prototype est expérimental et comprends des parties logiciels (simulation et code Arduino) et matériels (périphériques divers). La simulation a été codée à l'aide de du programme Unity et ensuite liée au composantes physiques du prototype à l'aide d'une carte et d'un programme Arduino. Il est à noter que les périphériques du produit final sont présents dans le prototype 2 mais qu'ils ne sont pas installés de manière permanente. Dans le prototype final, toutes les commandes seront installés dans une console. Par exemple, les boutons pourront être activés mais ne seront pas installés dans une console, mais plutôt libre sur une table. Pour cette raison, le budget de ce livrable reste presque nulle comme mentionné dans le plan des dépenses, puisque les composants utilisés pour faire ce prototype sont tous gratuites ou présentes dans le prototype final. Pour cette raison, ces dépenses en question ont été calculés dans le budget pour le prototype final.





```
TractorSimArduinoCommandBoard
//
}

encInewPins = 0;
encInewPins = (digitalRead(encInpinA)<<1|digitalRead(encInpinB));
encImove = RotaryDecodeTable[encIoldPins][encInewPins];
encIoldPins = encInewPins;
if (encImove == B01){
  if (counter < maxRotationPos){
    counter ++;
  }
}
if (encImove == B10){
  if (counter > maxRotationNeg){
    counter --;
  }
}

if (counter != lastAngle){
  Serial.println(counter);
}

lastAngle = counter;
delay(10);
}

int recvSerial() {
  if (Serial.available()) {
    int serialData = Serial.read();
    switch (serialData) {
      case '1':
        return 1;
        break;
      case '0':
        return 0;
        break;
      default:
        return -1;
    }
  }
}
}

1
Arduino/Genuino Uno on /dev/cu.usbmodem14601
```

```
TractorSimArduinoCommandBoard | Arduino 1.8.10

void loop ()
{
  if ((millis()-currentTime)> 125){
    int Blue, Green, Red, Yellow, light;
    double y, mov;
    y = analogRead (JoyStick_Y);
    light = digitalRead (JoyStick_Z);
    if (y < 515){
      y = -(518-y)/518;
      y = (y*10)/10;
      Serial.println(y,Z);
    }
    else if (y > 518){
      y = ((-(518-y))-12)/518;
      y = (y*10)/10;
      Serial.println(y,Z);
    }
    else{
    }
  }

  Blue = analogRead (BlueBot);
  Green = analogRead (GreenBot);
  Red = analogRead (RedBot);
  Yellow = analogRead (YellowBot);

  if (light == 0){
    digitalWrite(BlueLed, HIGH);
    digitalWrite(GreenLed, HIGH);
    digitalWrite(RedLed, HIGH);
    digitalWrite(YellowLed, HIGH);
  } else{
    digitalWrite(BlueLed, LOW);
    digitalWrite(GreenLed, LOW);
    digitalWrite(RedLed, LOW);
    digitalWrite(YellowLed, LOW);
  }
  if (Blue == 0){
    Serial.println("Blue Pressed");
  }
  if (Green == 0){
    Serial.println("Green Pressed");
  }
}

```

```

TractorSimArduinoCommandBoard | Arduino 1.8.10

TractorSimArduinoCommandBoard

void setup ()
{
  currentTime = millis();
  pinMode (JoyStick_Y, INPUT);
  pinMode (JoyStick_Z, INPUT_PULLUP);

  pinMode(enc1pinA, INPUT);
  pinMode(enc1pinB, INPUT);
  pinMode(enc1pinA, INPUT_PULLUP);
  pinMode(enc1pinB, INPUT_PULLUP);
  delay(1);
  enc1oldPins = (digitalRead(enc1pinA)<<1|digitalRead(enc1pinB));

  pinMode (BlueBoi, INPUT);
  pinMode (GreenBoi, INPUT);
  pinMode (RedBoi, INPUT);
  pinMode (YellowBoi, INPUT);

  pinMode(BlueLed, OUTPUT);
  pinMode(GreenLed, OUTPUT);
  pinMode(RedLed, OUTPUT);
  pinMode(YellowLed, OUTPUT);

  Serial.begin (9600);
}
void loop ()
{
  if ((millis()-currentTime)> 125){
    int Blue, Green, Red, Yellow, light;
    double y, mov;
    y = analogRead (JoyStick_Y);
    light = digitalRead (JoyStick_Z);
    if (y < 515){
      y = (518-y)/518;
      y = (y*10)/10;
      Serial.println(y,2);
    }
    else if (y > 518){
      y = ((518-y)-12)/518;
      y = (y*10)/10;
      Serial.println(v,2);
    }
  }
}

```

1

Arduino/Genuino Uno on /dev/cu.usbmodem14601

```

Project
  Scripts
    DS_Store
    ArduinoControls
    ArduinoControls.meta
    ArduinoTests
    ArduinoTests.meta
    CameraFollowController.cs
    CameraFollowController.cs.meta
    ComTests
    ComTests.meta
    SimpleCameraController.cs
    SimpleCameraController.cs.meta
    SimpleTractorController.cs
    SimpleTractorController.cs.meta
    SimpleWheel.cs
    SimpleWheel.cs.meta
    TractorController.cs
    TractorController.cs.meta

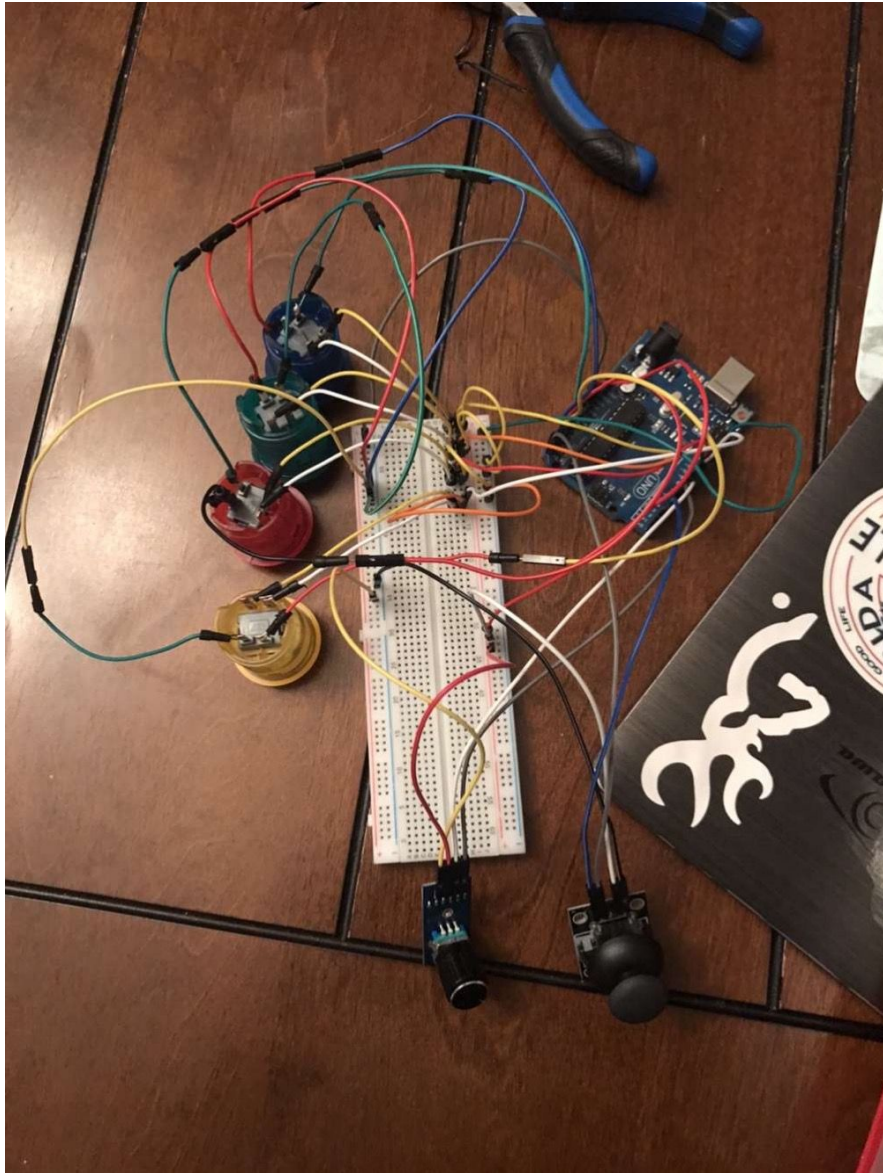
ArduinoControl.cs
13 public static int buttonState = 0;
14 int Timer = 0;
15 string leverPos;
16
17 void Start()
18 {
19   arduino.Open();
20 }
21
22 void Update()
23 {
24   if (Timer == 30){
25     string value = arduino.ReadLine();
26     buttonState = int.Parse(value);
27     Timer = 0;
28   }else{
29     Timer++;
30   }
31
32   if (buttonState == 1){
33     leverPos = "Forward";
34   }
35   else if (buttonState == 0){
36     leverPos = "Neutral";
37   }
38   else if (buttonState == -1){
39     leverPos = "Reverse";
40   }
41 }
42
43 void OnGUI()
44 {
45   string newString = "Connected: " + leverPos;
46   GUI.Label(new Rect(10,10,300,100), newString);
47   {
48     // Register the window. Notice the 3rd parameter
49     windowRect = GUI.Window(0, windowRect, DoMyWindow, "My
50   }
51 }
52

TractorController.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using System;
5
6 //AUTHOR @Hubert Lachaine//
7
8 public class TractorController : MonoBehaviour {
9
10  public void GetInput()
11  {
12    m_horizontalInput = Input.GetAxis("Horizontal");
13    m_verticalInput = ArduinoTest.buttonState;
14  }
15
16  private void Steer()
17  {
18    m_steeringAngle = maxSteerAngle * m_horizontalInput;
19    frontDriverW.steerAngle = m_steeringAngle;
20    frontPassengerW.steerAngle = m_steeringAngle;
21  }
22
23  private void Accelerate()
24  {
25
26    frontDriverW.motorTorque = m_verticalInput * motorForce;
27    frontPassengerW.motorTorque = m_verticalInput * motorForce
28  }
29
30
31  private void UpdateWheelPoses()
32  {
33    UpdateWheelPose(frontDriverW, frontDriverT);
34    UpdateWheelPose(frontPassengerW, frontPassengerT);
35    UpdateWheelPose(rearDriverW, rearDriverT);
36    UpdateWheelPose(rearPassengerW, rearPassengerT);
37  }
38
39  private void UpdateWheelPose(WheelCollider _collider, Transf
40  {

```

ArduinoTest.cs 11

CRLF UTF-8 C# GitHub Git (0)



Points d'amélioration pour les prototypes (1 et 2)

Après la dernière rencontre avec le client, la rétroaction du prototype 1 fut très utile et pertinente. Notre prototype 1 semblait très complet, et les aspects ergonomiques des composantes étaient très bonnes. Par contre notre cliente nous a fait remarquer de garder les boutons simples, et de limiter la quantité de ceux-ci. Ceci fut implémenté lors du deuxième prototype, puisque la simulation ne prend que quatre boutons, soit une diminution de 9 boutons, du prototype 1 au prototype 2. La disposition des boutons et leviers sera également modifiée pour le prototype final suite à des commentaires faits par la cliente au niveau de l'accessibilité des périphériques. Pour ce qui en est du deuxième prototype, la rétroaction est encore à venir donc les points à améliorer ne sont pas encore définis. Pour cette raison, ces derniers seront absents de ce livrable. De plus, nous n'aurons pas la chance de rencontrer les clients pour recevoir leur rétroaction sur le prototype 2. C'est pour cette raison que nous devons nous fier sur les contraintes imposées par les clients au début du projet ainsi que notre propre rétroaction personnelle après avoir analysé le prototype en équipe.

Conclusion

Pour conclure, le prototype 2 consistera de notre simulation codée avec le logiciel Unity et du système Arduino (du côté logiciel ET physique) pour enregistrer les commandes de l'utilisateur et ensuite les transmettre à l'écran du simulateur. À ce stade du projet, avec la rétroaction du client sur le prototype 1 ainsi que notre propre rétroaction du prototype 2, nous serons en mesure de compléter le prototype 3 en respectant toutes les contraintes imposées. Il reste encore à concevoir le prototype 3 en intégrant le prototype 2 à une version améliorée du prototype 1 pour avoir un prototype final complet.