Printer I/O Game System

November 2018

Can Ur, Wessel Mast

Game System / Game Platform

Table of contents

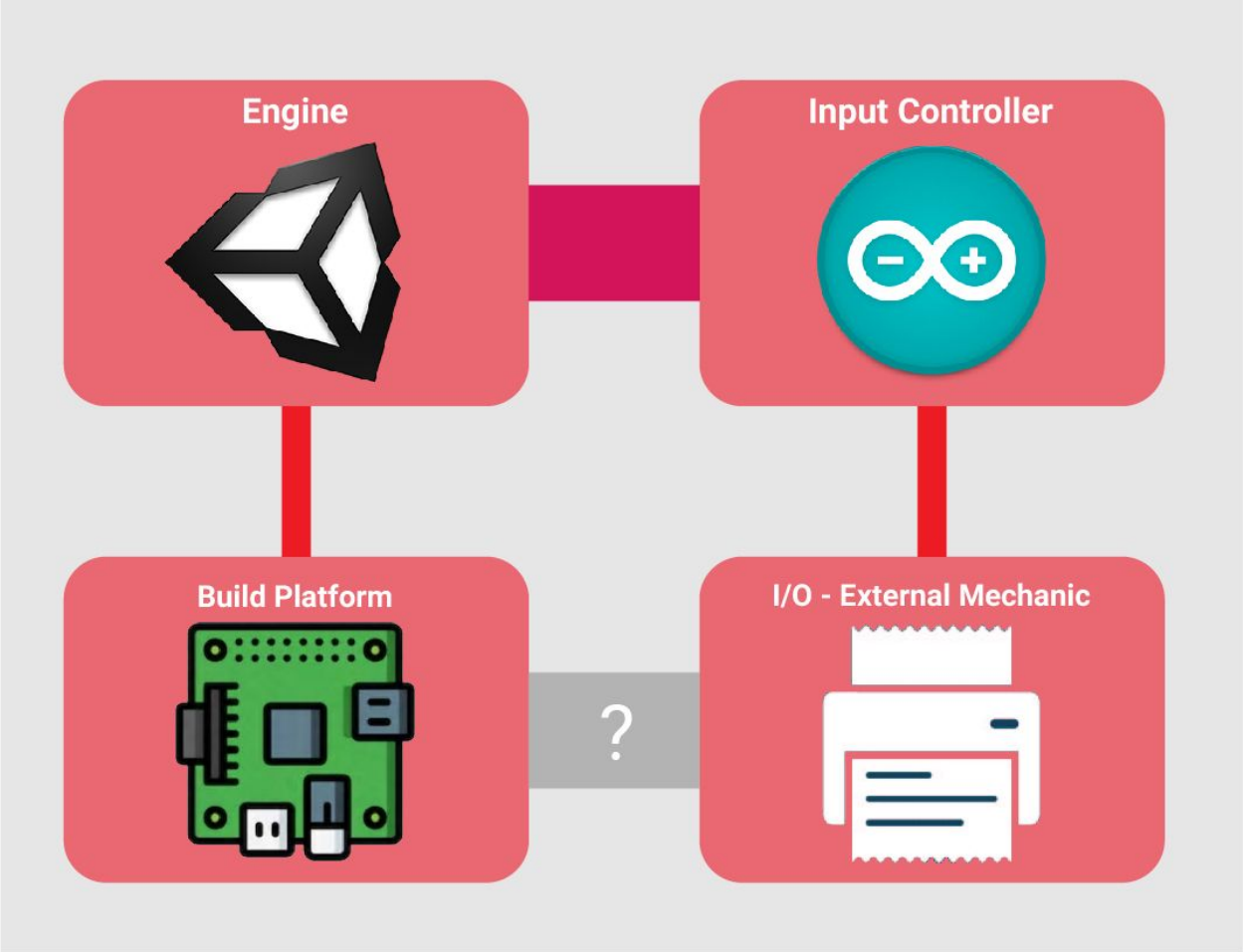
Concept

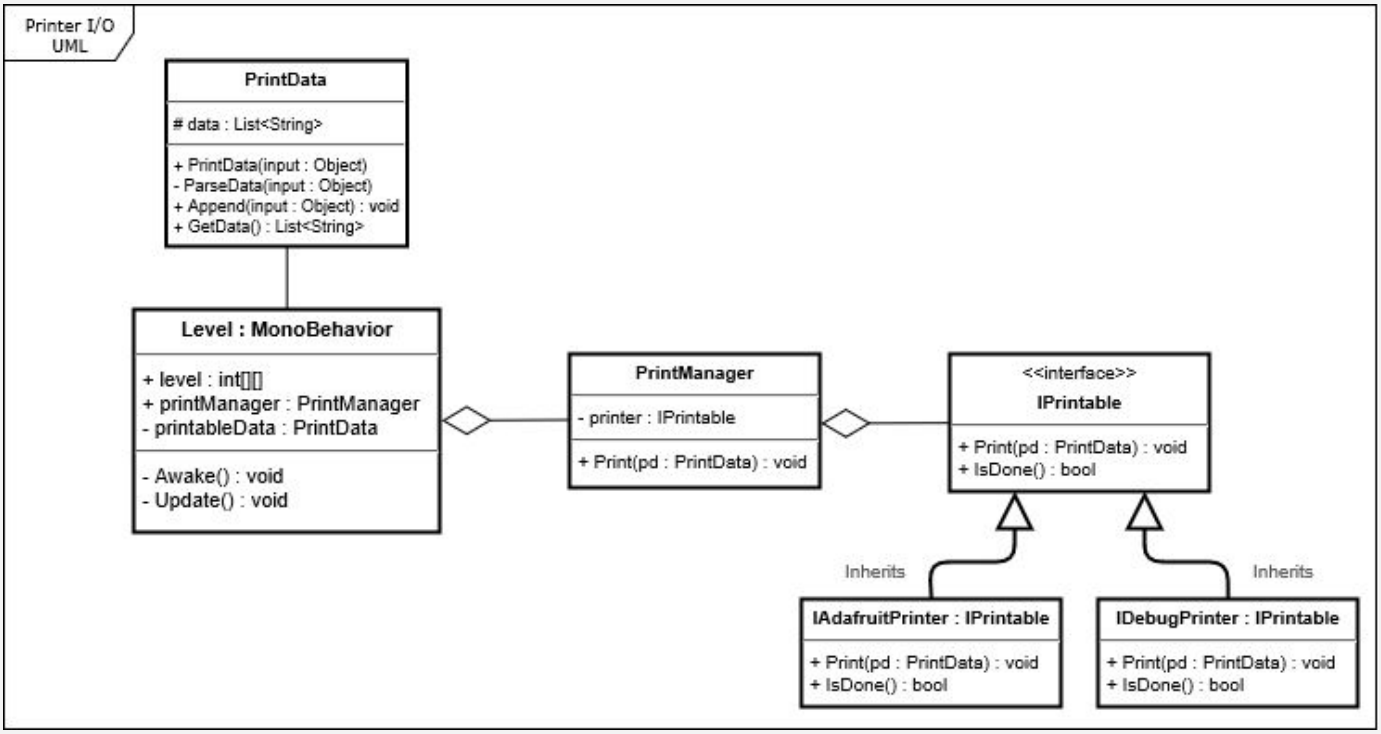
This project is a multiplatform game system which combines the digital interaction of gaming, with the physical aspects of printing technology.

The system will achieve platform agnosticism by separating this printing part of the main core system, and being modular / extendable.

The first iteration of the system design (26-11-2018)

Rectangular Communication system **(seen below)**

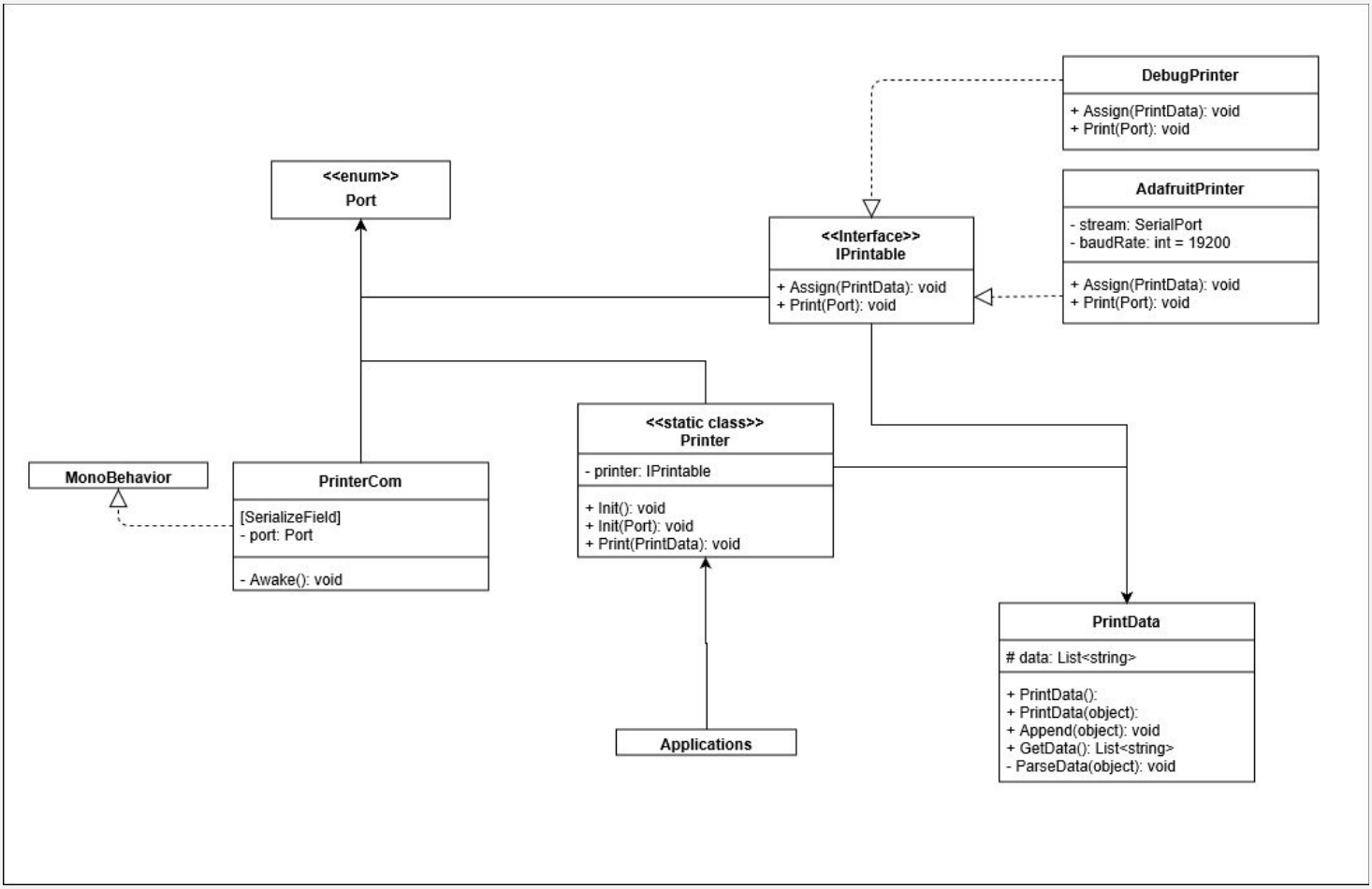




First iteration UML diagram

The system started as a Printing **and Scanning** platform, but due to time constraints and lack of efficiency we chose to focus solely on the Printing system communication with the main system. Setting this as our primary goal (Game- and Print System efficiency) also eliminated other aspects of our initial design, such as the role of the Raspberry PI platform (Build platform). This did not seem like a requisite in our main concept, but rather an unnecessary extension that would only act as the main hub for the system and not change much of relevance.

Putting this aside, a couple of system iterations have been developed and we have achieved a simple but efficient form of the concept. And overall; It went quite well and we both improved in how to tackle a platform agnostic system design.



Final version of the UML diagram

The system, as it is now, can be dissected into three groups.
- Applications
- The Main System
- The Printing Handler

These subsections are all separated of each other, and the communication within these parts have limited entry (through functions). This provides an organised and clear way of problem solving, and eventually tracking the sent data. Games and applications are on the user-end of the spectrum; They can create the data and only send it to the printer through the Main System, which handles the other hardware-connected parts in the background.

Technical Design / Code Architecture

- From Dependency Injection to precompiled command C# dinges, automated system recognizing #aaron Degenerate tangent

- The IPrintable interface defines the fundamental functions a printer library would need to implement, in order to work.

- We make use of a Facade pattern in PrintManager by handling all the low-level printing operations behind the curtains; The user or surface application developer will barely get a glimpse of what is executing in the background. We wanted the surface code (for the applications / games, the MonoBehavior) to be as small and simple as possible.
- The main system uses precompiled headers in the monobehavior to decide what the program has to execute in certain situations, on certain platforms. In this instance, on builds the Adafruit printer is assigned as the main printing unit, but in the Unity editor the Debug Console is assigned as the primary unit.
- Printer is a public static class that gives any script the ability to send print commands and start the printing process in the hierarchy.
- PrintData.cs has the ParseData function that is able to translate any value of a lot of common variable types to a printable format.
- A custom namespace wrapper don't allow for classes that aren't meant to print to accidentally print.

- The specific Adafruit Printer can establish the printer connection by letting the user select a COM port.
An algorithm handles COM ports that are above COM9, preventing errors and further inconsistencies.

```
string[] portNums =
System.Text.RegularExpressions.Regex.Split(port.ToString(), @"\D+");
        stream = (int.Parse(portNums[1]) >= 10) ? new
SerialPort("\\\\.\\" + port.ToString(), baudRate) :
                                new SerialPort(port.ToString(), baudRate);
```

Game Concepts

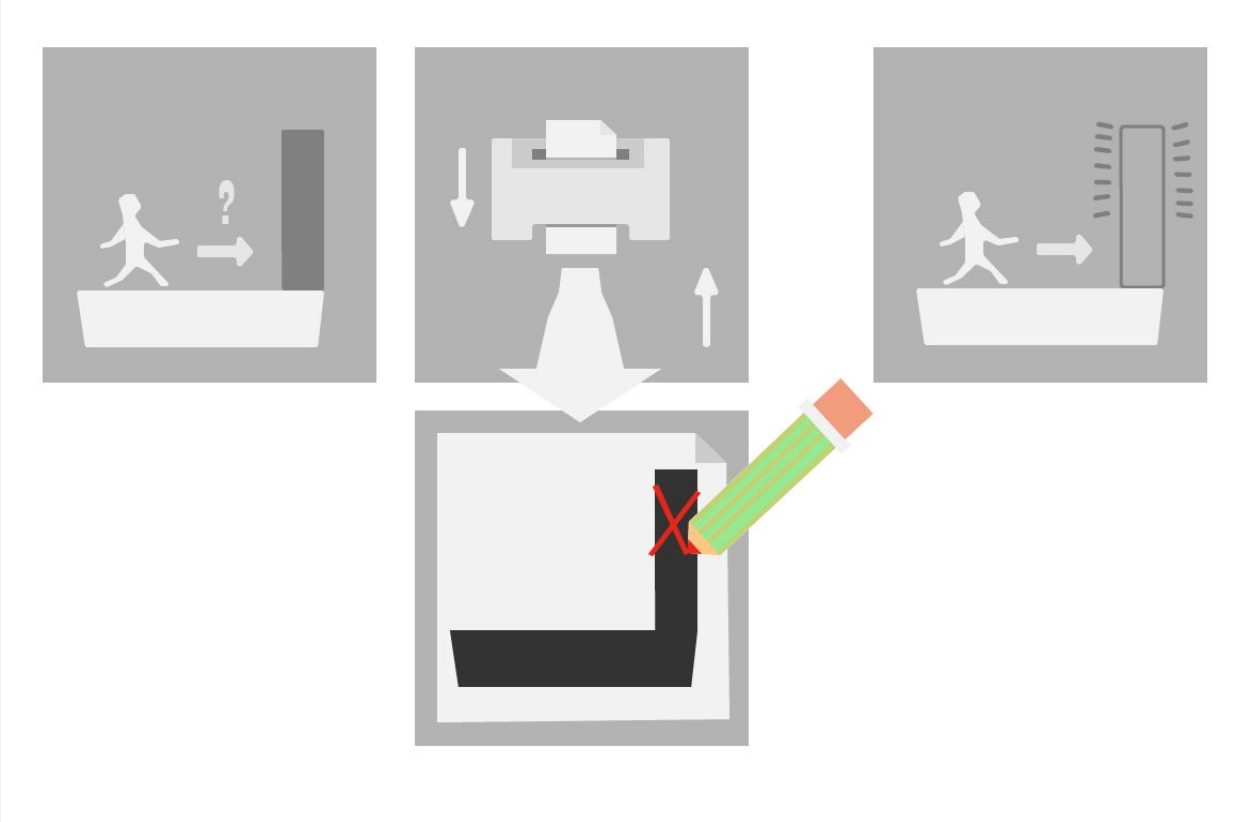
We came up with a couple of game concepts, that would use the printing system in an interesting way:

- Printing + Scanning user changes to the game world.
Scribblenauts, Drawn to Life

- Jumper game w/ vertical receipt printing of your playthrough + high score and point of death
Doodle Jump

- Roguelike w/ scores, highscore & Map seed as custom QR-like formatted visual code.
Spelunky? Binding of Isaac

- Draw on anything, and scan it in; It becomes the level you play on.
Line Rider



Printing + Scanning user changes to the game world (Idea #1)

Log

14-11-2018 (2 uur):

First concept idea

19-11-2018 (2 uur):

Reworked the concept. It now is more coherent in its design, and the goals are more reachable. (Arduino, Raspberry PI, Printer & Unity). Thermal-printing research.

26-11-2018 (2 uur):

Planning + Concept idea made concrete.
Gathered materials for building the system.

03-12-2018 (5 uur):

Development and first iteration of the code architecture.
The system is built around abstractions of interfaces, and a Facade pattern to handle complex functions of communication between the systems (The printer, a potential scanner). This Facade ultimately connects the Level with the printable Data structure.

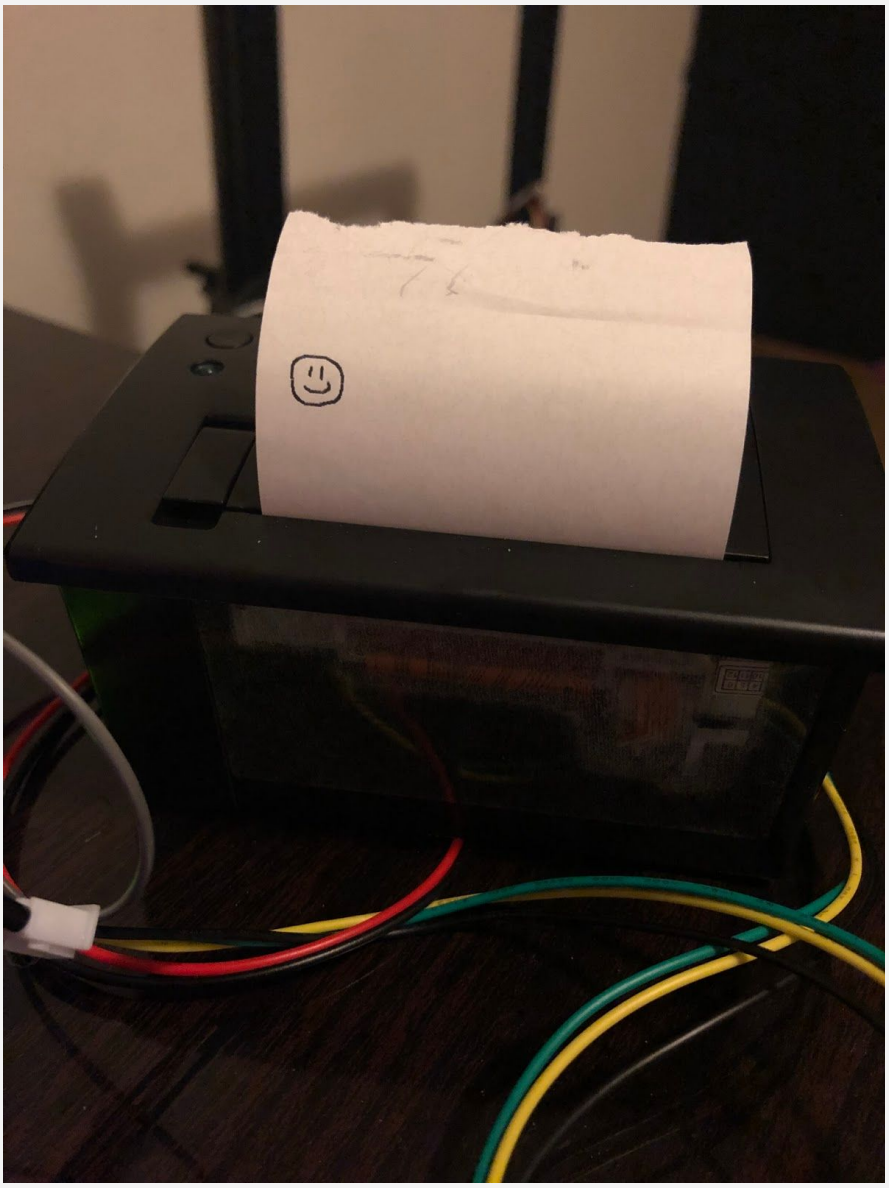
The PrintData class handles the conversion of all the basic data types (Float, Int, Double, Char) to a printable format, and also has some workarounds for converting Lists and Arrays into a usable standard.

04-12-2018 (4 uur):

Scanner Research + Code Architecture UML Chart first iteration

10-12-2018 (5 uur):

First test & experimentation with the Adafruit Thermal Mini-printer and Arduino. Used the provided printing library which supports text in three sizes (S, M and L), barcodes and 1-bit monochrome bitmap images. Encountered some bugs and limitations of the system, misprinting the bigger font sizes. (Probably due to lower voltage / current than recommended). Future testing will be needed.
Also got Serial Communication working between Unity and Arduino.



11-12-2018 (4 uur):

Developed an application with a textfield UI element, which allows the user to communicate through the system architecture with the printer, and directly print a string on a receipt.

26-12-2018 (5 uur):

Developed a second application which was a small painting app. The idea here was that users could draw an image and print it. For now the app contained just a brush size slider a brush and an eraser.

28-12-2018 (4 uur):

Further developed the painting application, working on encoding to a bmp image which could then be used in the LCD Assistant application to convert it to a byte array that could be used to print images. This didn't succeed today. What did succeed was tons of refactoring and reworking the system to feel much better and to be structured better.

06-01-2018 (9 uur):

Working more on that encoder. Lots more tears were shed today as it ended up being harder than it seemed. Also, no-one seems to have shared trying this before, so the internet was no help. Also recorded footage and edited the video.

08-01-2019 (6 uur):

Optimisation, merging of Canvas Drawer application with PrintScreen text printer app.
Added COM-port selection in editor / detection in code if printer is attached, otherwise revert to the default Debug.Log-Printer.

18-01-2019 (2,5 uur):

Finalisation of the project, updated UML diagram + Documentation. *Final reflective comments on this project.*

Inspirations

https://learn.adafruit.com/mini-thermal-receipt-printer/microcontroller

Research (Sources)

http://gieskes.nl/instruments/?file=image-scan-seq
https://www.youtube.com/watch?v=I5spdfnEZeA

https://thepihut.com/products/pixy-1-0-vision-sensor-for-the-arduino?variant=796390889

https://www.explainthatstuff.com/barcodescanners.html
https://www.explainthatstuff.com/how-photoelectric-cells-work.html

Arduino & Unity Serial Communication:
https://medium.com/@yifeiyin/communication-between-arduino-and-unity-9fdccc2be3f

https://www.alanzucconi.com/2015/10/07/how-to-integrate-arduino-with-unity/

Printing Adafruit Tutorial:
https://learn.adafruit.com/mini-thermal-receipt-printer/printing-text