gym KIRCHENFELD

Der Bau eines Arcade Cabinets

Von Dario Strübin

Betreut von Sebastian Forster

Gymnasium Kirchenfeld

Abteilung MN

2014

Inhalt

1. Einleitung

1.1 Was ist ein Arcade Cabinet?

Eine der grössten Schwierigkeiten, welcher ich im Verlaufe des Projektes begegnete, war es, den Leuten zu erklären, was ein Arcade Cabinet denn eigentlich ist. Das Fehlen eines passenden deutschen Wortes erschwerte die Situation weiter und meistens kam ich nicht drum herum, ein Bild hervor zu nehmen, um die Frage zu klären.

Arcade Cabinets sind grob gesagt Spielkästen, auf denen Videospiele gespielt werden können. Sie verbreiteten sich besonders stark in den späten 70ern, einer Zeit, in der Videospiele einzig auf solchen Automaten spielbar waren. Durch die Verbreitung von Computern und Heimkonsolen wurden sie jedoch bald verdrängt und sind heute nur noch selten anzutreffen. Gleichwohl wurden in jener Zeit viele Spiele geschaffen, die Klassiker wurden und bis heute bekannt sind. Beispiele dafür sind etwa Pong, Space Invaders oder Pac-Man.

1.2 Motivation & Idee

Gerade in einer Zeit, in welcher die Videospiele zunehmend an Komplexität gewinnen und sich ihre Grafik immer mehr dem Fotorealismus nähert, findet man wieder vermehrt Gefallen am Charme der simplen, alten Arcade Games. Mit Emulatoren sind diese auch am Computer spielbar und als Hobby-Programmierer versuchte ich auch selbst, solche altmodischen Spiele neu zu schreiben. Ich kam jedoch zu der Erkenntnis, dass durch das Ersetzen des Automaten mit einer Maus und einer Tastatur ein essenzielles Element verloren geht, ohne welches das ursprüngliche Spielerlebnis nicht reproduzierbar bleibt. Um mit den Arcade-Spielen den gewünschten Effekt zu erzielen, fasste ich also den Entschluss, ein Arcade Cabinet zu bauen.

1.3 Aufteilung der Arbeit in verschiedene Bereiche

Die hohe Anzahl von Aufgaben, aus welchen der Herstellungsprozess bestand, wurde für die schriftliche Arbeit in die folgenden drei Bereiche aufgeteilt:

Kastenbau:	Alles was die Herstellung des eigentlichen Kabinettes betrifft: Das Kapitel wurde in zwei weitere Teile aufgeteilt. Im ersten geht es um das Erstellen der Pläne und das Zurechtschneiden der Bauteile. Der zweite Teil behandelt dann die Montage und die Dekoration des Kastens.
Elektronik:	Fokus auf die Schnittstelle zwischen Mensch und Computer: Die Eingaben, welche auf Knöpfen und Joysticks gemacht werden, müssen in ein digitales Format umgewandelt werden, welche vom Computer eingelesen werden können. Ein weiterer Teil dieses Kapitels behandelt die Integration von Lautsprechern.
Informatik:	In diesem Bereich geht es um das Programmieren der Software und der Videospiele.

2. Kastenbau Teil 1: Erstellung der Pläne und Bauteile

2.1 Virtuelles Gestalten des Kabinetts

Die Gestaltung und Planung des Arcade Cabinets geschah zum grössten Teil virtuell, mithilfe von 3D-Computergrafik- und CAD-Programmen. Dadurch konnte der Kasten komplett nach den eigenen Vorstellungen und Wünschen entworfen werden. Im ersten Schritt wurde ein dreidimensionales Modell mit der graphischen Software 3DS Max gestaltet. Die Masse und Dimensionen dieses ersten Modells stimmten nur grob und waren nicht geeignet, um die Teile zuzuschneiden. Wegen der grossen Flexibilität und der Möglichkeit, das Kabinett einfach und ansprechend zu visualisieren, war 3DS Max jedoch eine gute Wahl, um den Kasten zu entwerfen und mit verschiedenen gestalterischen Möglichkeiten zu experimentieren.

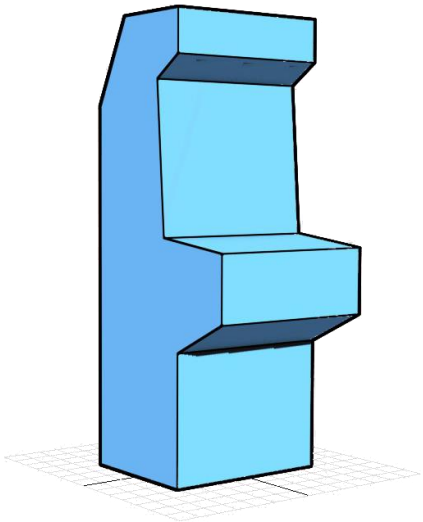


Abb. 1: So sah die erste Version des Modells aus. Sie wurde natürlich noch stark abgeändert und konkretisiert, weist aber dennoch die Grundform auf.

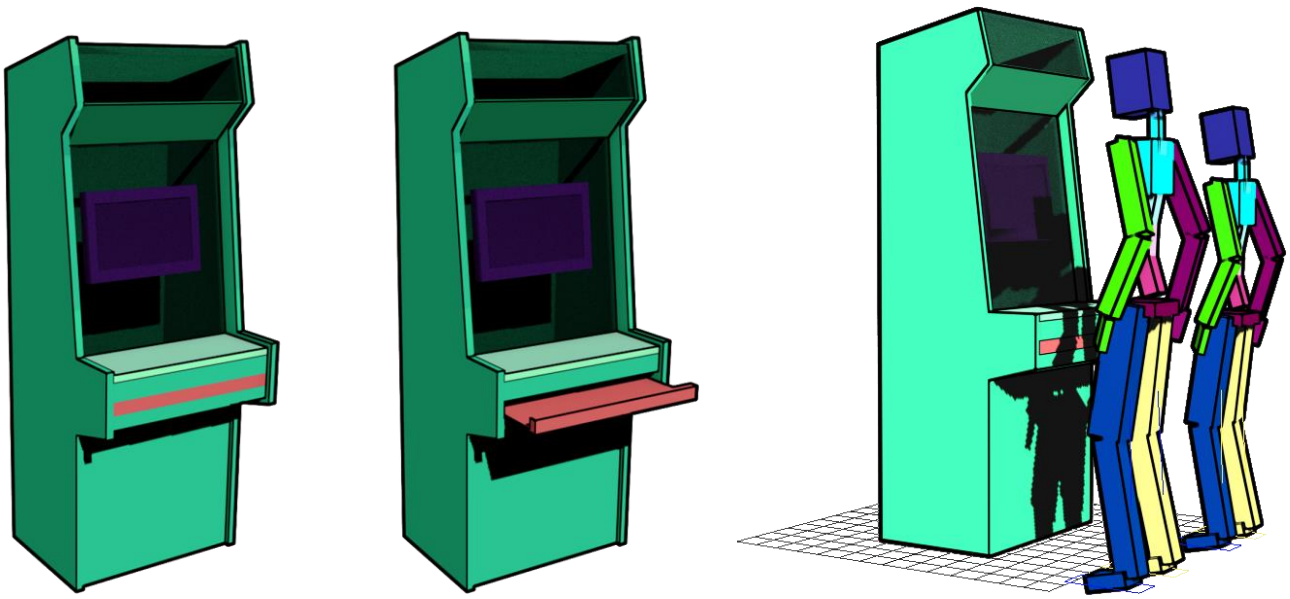


Abb. 2 und 3: Renderings der Endversion des grafischen Modells.

2.2 Berechnung und Zeichnung der Pläne

Um genaue und flexible Baupläne erstellen zu können, wurden die exakten Zeichnungen ebenfalls digital gemacht. Dieses zweite virtuelle Modell wurde mit der Software NX von Siemens gezeichnet, welche anders als 3DS Max ermöglicht, rechnerisch präzis jedes einzelne Bauteil zu dokumentieren. Da NX spezifisch für Ingenieure und generell Branchen im Produktionsbereich ausgerichtet ist, kann es die Konstruktionen und Formen so exportieren, dass sie beispielsweise von einem 3D-Drucker gedruckt werden können. Dies war der zweite Grund für die Rekonstruktion des Kabinettes in dieser Software, denn es wurde später in der Herstellung der Bauteile von einer Maschine Gebrauch gemacht, welche die komplizierteren Stücke automatisch aus einer Holzplatte fräste. Dafür wurde aber zwingend ein Dateiformat erfordert, welches 3DS Max nicht ausgeben konnte.

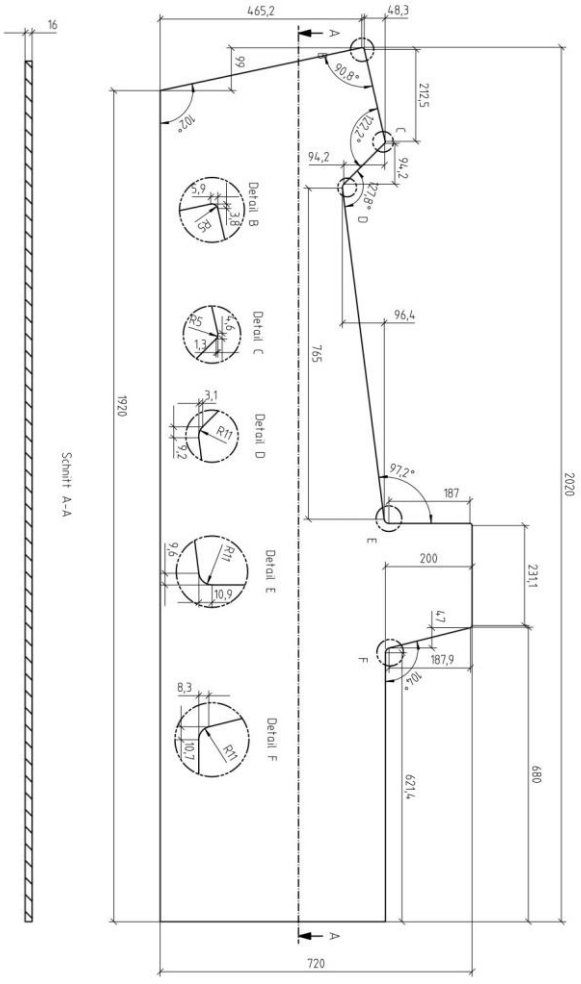


Abb. 4: Bauplan der Seitenwände

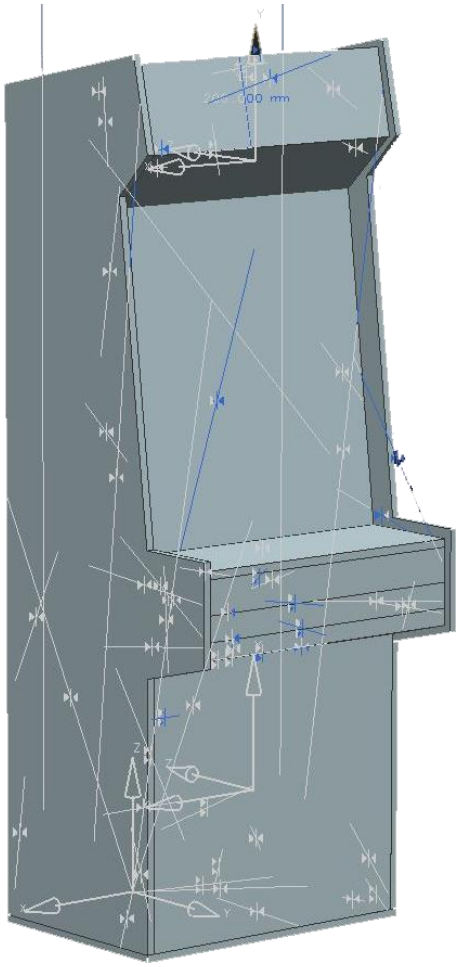


Abb. 5: Das fertige NX-Modell

2.3 Herstellung der Bauelemente

Mit den fertigen Bauplänen war nun alles bereit, um zur Produktion voranzuschreiten. Um die genauen Zeichnungen aber umzusetzen und ihr Potential auszuschöpfen, hätten eine herkömmliche Säge und etwas Schleifpapier kaum ausgereicht. Ihre Realisierung wurde jedoch durch die Schweizer Möbelfirma Girsberger ermöglicht, welche grosszügig die nötigen Arbeitsmaschinen zur Verfügung stellte und bei der Beschaffung des Holzes half.

Für das ganze Kabinett wurde mitteldichte Holzfaserplatte mit 16mm Dicke verwendet. Dies ist ein Material, welches gute Stabilität bringt, sich gut anstreichen lässt und zu einem günstigen Preis erhältlich ist.

Die beiden Seitenwände waren die mit Abstand komplexesten Stücke und stellten anfangs ein Problem für die Herstellung dar. Sie sind über zwei Meter hoch und bis zu 72cm breit und mussten je aus einem einzelnen Stück gefertigt werden. Dazu kommt, dass sie eine Form mit Vorsprüngen und Rundungen haben und die massgebende Gestalt des Kabinettes prägen. Um diese Herausforderung zu bewältigen, wurde von einer von Girsbergers besten Maschinen Gebrauch gemacht, welche das CAD-Modell aus dem NX einlesen konnte und die gewünschte Form vollkommen automatisch aus einer Holzplatte herausschneiden konnte.

Die restlichen Stücke haben alle eine rechteckige Form und waren somit wesentlich simpler herzustellen. In einem ersten Schritt wurden sie alle in die richtigen Dimensionen zugeschnitten und dann im zweiten Schritt, falls nötig, an der Kante angewinkelt.

Abb. 6: Hinter dieser Maschine werden gerade die beiden Seitenteile nach Plan zugeschnitten

3. Elektronik

3.1 Wie der Computer einen Knopfdruck erkennt

Eine der grössten offenen Fragen, die sich stellte, als ich die Arbeit begann, war, wie Knopfdrücke und Joystick-Bewegungen am Computer eingelesen werden sollten. Beide Eingabegeräte haben im Prinzip eine sehr simple Funktionsweise. Wird ein Taster in einen Stromkreis eingebaut, so wird dieser bei einem Knopfdruck geschlossen und der Strom fliesst. Ein Joystick funktioniert auf die genau gleiche Weise, wobei es hier für alle vier Richtungen einen Anschluss gibt. Das Problem an dieser Stelle ist, wie man in der Software einlesen kann, ob ein Stromkreis geschlossen ist oder nicht. Schliesslich besitzen Computer keine solchen Schnittstellen.

Die Lösung zum Problem ist der „Arduino“, ein kreditkartengrosser Mikrocontroller mit programmierbaren Pins. Diese Pins können dann entweder eine Spannung messen oder selbst eine anlegen. Dazu kann der Arduino über USB eine serielle Schnittstelle mit dem Computer aufbauen und somit die erhaltenen Daten senden.

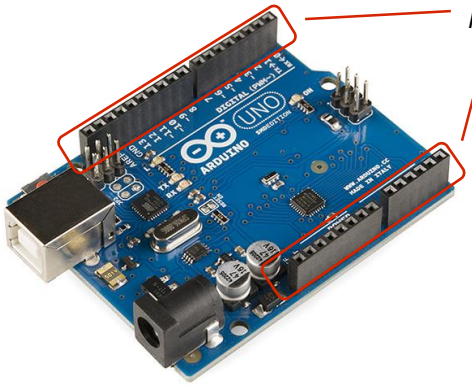


Abb. 7: Bild eines Arduinos

3.2 Schaltkreis eines Knopfes

Um die Eingaben einzulesen, wurde zuerst schlicht ein Plus-Pol mit 5 Volt an einen programmierbaren Pin angeschlossen und dazwischen ein Knopf gesetzt. Dieser Pin wurde dann in die „Lesefunktion“ gesetzt, was heisst, dass er schaut, ob eine Spannung vorhanden ist. Ist das der Fall, so gibt er eine Eins aus, ansonsten eine Null. Das Ergebnis fiel jedoch nicht wie erwartet aus. Wurde der Knopf gedrückt, so resultierte die erwartete Eins, jedoch erhielt man wenn nichts gedrückt wurde eine zufällige Fliesskommazahl im Bereich zwischen 0 und 1. Der Grund dafür ist, dass der Arduino, wenn er nichts messen kann, dies nicht mit einer Null gleichsetzt, sondern nach dem Prinzip von Schrödingers Katze alle Möglichkeiten dazwischen ausgibt.

Ist der Taster also nicht gedrückt, so muss der Minus-Pol angeschlossen sein. Damit beim Knopfdruck kein Kurzschluss entsteht und der Pin noch eine Spannung messen kann, braucht es davor noch einen Widerstand.

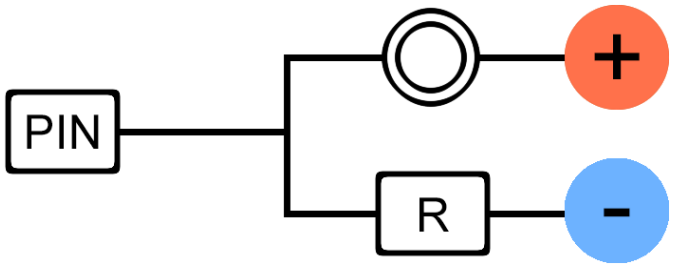


Abb. 8: Korrekter Schaltkreis eines Knopfes

3.3 Inputerweiterung durch Schieberegister

Ein Problem, welches der Arduino mit sich bringt, ist die begrenzte Anzahl an Pins. Um mich nicht davon einschränken zu lassen, suchte ich nach einer Möglichkeit diese zu erweitern. Der Einsatz von mehreren Arduinos war ausgeschlossen, da dies umständlich und kompliziert für den Datenaustausch, teurer und einfach generell keine „schöne" Lösung gewesen wäre. Ich stiess also nach etwas Recherche auf die „Schieberegister", welche genau den erwünschten Zweck erfüllten.

Abb. 9: Das SN74HC165N Schieberegister von Texas Instruments

Der SN74HC165N ist ein Schieberegister mit acht parallelen Eingängen, welche jeweils wie ein Pin mit einem Knopf oder Joystick verbunden werden können. Dann besitzt er fünf weitere Ein/Ausgänge, welche mit Pins am Arduino verbunden werden müssen. Davon zwei für die beiden elektrischen Pole, zwei für die Steuerung und der letzte für den Datenausgang. Ist alles verbunden, so werden in regelmässigen Abständen die Daten von den acht Eingängen eingelesen. Dazu löst der Arduino einen Steuerimpuls, einen sogenannten „Shift" aus. Das vorderste Bit wird dadurch zum Arduino geschoben und dort registriert. Alle anderen Bits werden aber ebenfalls eine Stelle nach vorne gerückt und der Prozess wiederholt sich, bis alle durch sind.

Mit dieser Methode konnten jetzt aus fünf Pins am Arduino acht gemacht werden, was schon mal besser, aber noch nicht besonders beeindruckend war. Doch damit war das Limit nicht erreicht, denn Schieberegister können auch in Serie geschalten werden und somit die Anzahl an Eingängen erhöhen, ohne mehr Pins zu brauchen. Dabei wird bei einem Shift das vorderste Bit des zweiten Schieberegisters in den hintersten Platz des ersten geschoben. Für dieses Projekt reichte es, zwei Stück hintereinander zu schalten und somit 16 Eingänge zu erhalten. Es wäre aber auch möglich gewesen, mit derselben Technik noch viele weitere anzuhängen.

3.4 Fertigstellung des Spielbretts

Da die Theorie für das Einlesen der Eingabegeräte soweit gelöst war, galt es, diese umzusetzen. Ich begann damit, die Elektronik zur Inputeinlesung auf einer Steckplatine zu testen und die Schaltpläne dazu zu erstellen. Zum Schluss wurde dann alles an eine Epoxyd-Platine gelötet.

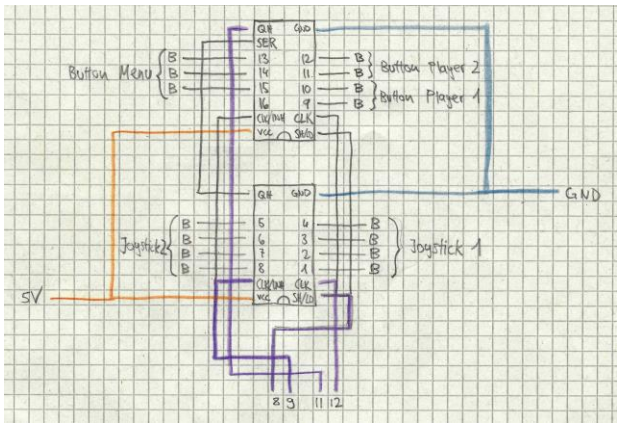


Abb. 10: Plan des fertigen Schaltkreises

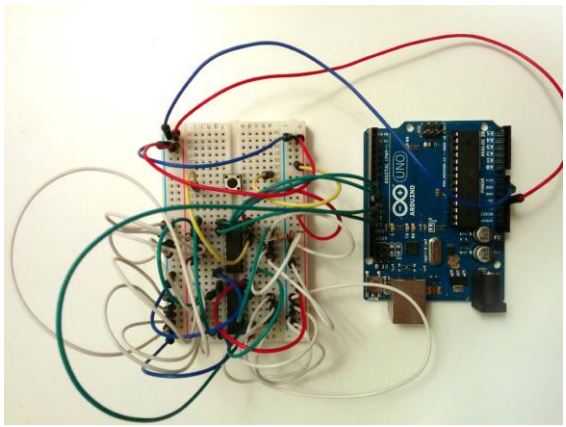


Abb. 11: Der fertige Schaltkreis an der Steckplatine

Als nächstes wurden alle Eingabegeräte an das Spielbrett montiert. Dieses ist so konzipiert, dass zwei Spieler jeweils mit einem Joystick und zwei Knöpfen spielen können. Dazu kommen drei grosse, rot leuchtende Menüknöpfe, um das Spiel zu pausieren, minimieren oder zu schliessen. Dann wurde die Platine mit dem Arduino an die Unterseite des Brettes montiert und an die Geräte geschlossen.

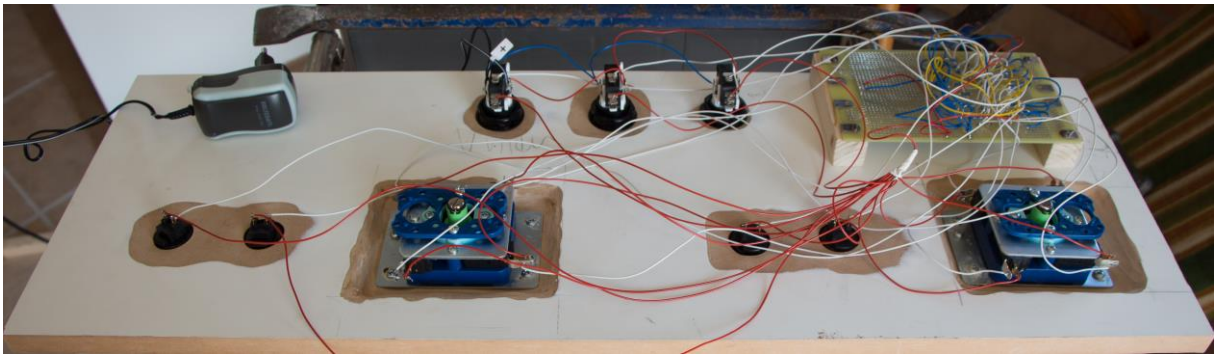


Abb. 12: Die Unterseite des fertigen Spielbrettes

3.5 Bau der Lautsprecher

Für die Lautsprecher wurde nach einer möglichst kosteneffizienten Lösung gesucht, da es ohnehin schwierig ist, selber guten Sound in die Spiele einzubauen. Dazu wurden zwei kleine Lautsprecherchassis an das Brett oberhalb des Bildschirmes montiert und an einen kleinen Verstärker gebunden. Dazu wurde ein kleiner Schalter miteingebaut, um die Lautsprecher ein-, bzw. auszuschalten. Das ganze Modul kann dann simpel über einen 3.5mm Klinkenstecker an den Computer angeschlossen werden.

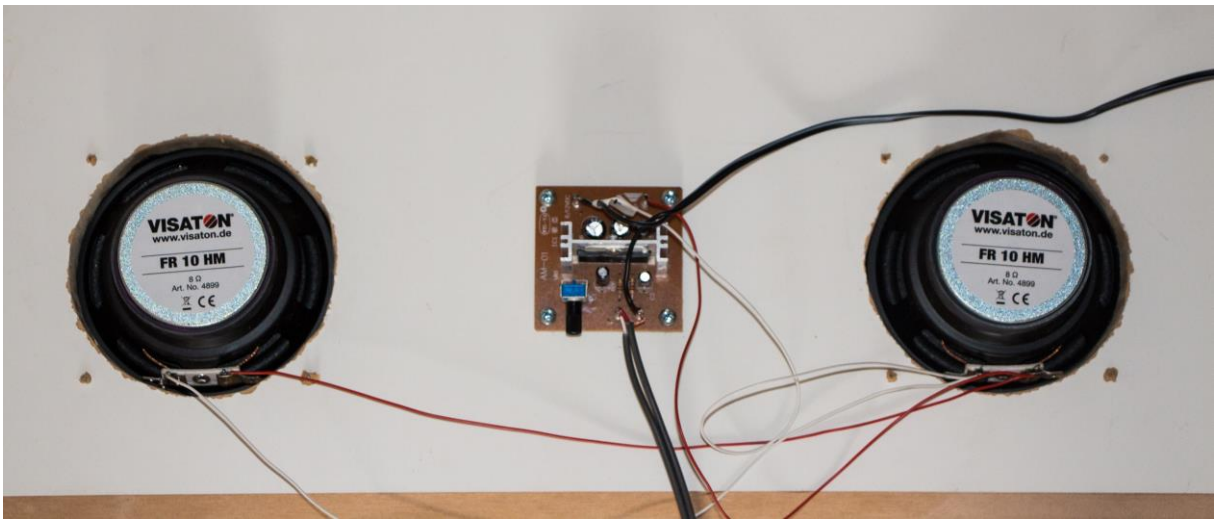


Abb. 13: Rückseite der Lautsprecher. In der Mitte das Verstärkermodul.

4. Kastenbau Teil 2: Montage und Dekoration

4.1 Montage des Kabinetts

Die Teile waren nun zugeschnitten, befanden sich im Atelier und waren mit der Elektronik bestückt. Das Arcade Cabinet konnte nun endgültig zusammengebaut werden.

Das Ziel war es, so weit wie möglich die Schrauben von innen einzuführen, um möglichst wenig Spuren des Baus zu hinterlassen und die Ästhetik nicht zu stören. Dazu wurden schmale Holzlatten entlang der Kanten der beiden Seitenwände angebracht.

Abb. 14 und 15: Befestigungstechnik

Danach wurden als erstes die beiden Seitenwände auf die Grundplatte befestigt. Alle weiteren Bauteile wurden dann zwischen diese beiden Wände positioniert und von innen, wie auf der Abbildung sichtbar, an die Holzlatten geschraubt.

Abb. 16-18: In wenigen Schritten nimmt das Kabinett Form an.

Es kann vorkommen, dass der Computer, auf welchem das Arcade-System läuft, gewartet werden muss oder seine Software ein Update braucht. In solchen Fällen ist man darauf angewiesen, die herkömmlichen Eingabegeräte Maus und Tastatur benutzen zu können. Um diese jederzeit leicht zugänglich zu machen, wurde eine ausziehbare Schublade in der Mitte des Vorsprungs eingebaut.

Für den Bildschirm wurde eine Halterung gebaut, welche die richtigen Masse hatte, um im Kabinett angebracht zu werden.

Darauf mussten nur noch die Bretter mit den Spielkomponenten und den Lautsprechern eingebaut werden und es konnte bereits eine erste Zwischenbilanz gezogen werden. Der Computer wurde in das Innere des Kabinetts platziert und mit allem verbunden. Glücklicherweise hat auch alles gleich funktioniert und es konnten schon erste Spiele gespielt werden.

Abb. 19-21

4.2 Dekoration und Fertigstellung

Nachdem die Grundelemente des Kabinetts zusammengebaut waren, mussten noch ein paar letzte Aufgaben erledigt werden, bevor zur Bemalung vorgeschritten werden konnte.

Das erste bisher vernachlässigte Problem war, dass man den Computer, welcher sich innerhalb des Kabinettes befindet, irgendwie einschalten können muss. Um das zu lösen, wurden die Kabel innerhalb des Computers, welche vom Mainboard an den Startknopf gehen, gekappt und an einen eigenen Taster gelötet. Dieser wurde dann an den Vorsprung montiert. Zu beachten ist, dass der Startknopf so konfiguriert wurde, dass er den Computer nur einschalten kann, damit man das Arcade-System nicht ungewollt während dem Spielen herunterfährt.

Zuletzt wurden dann noch Griffe an die grosse Platte auf der Rückseite des Kabinettes montiert. Der Grund dafür ist, dass diese Platte auch nach der Verschliessung einfach entfernbar sein muss, um zum Computer oder allgemein der Elektronik Zugang zu haben. Aus demselben Grund, ist dieses Bauelement von aussen und nicht von innen angeschraubt.

Somit konnte ich mich nun der Bemalung zuwenden. In einem ersten Schritt wurde das gesamte Kabinett mit drei Schichten schwarzen, matten Lacks bestrichen. Danach wurden die Kanten der Beiden Seitenwände, sowie einiger weiterer Elemente mit blauem Lack eingefärbt.

Auf die Fläche unterhalb des Vorsprungs sollte das Arcade Logo mit derselben blauen Farbe gemalt werden. Dieses Logo, welches aus einem Alien aus dem Arcade-Klassiker Space Invaders und einem simplen Schriftzug besteht, wurde am Computer gestaltet, ausgedruckt und auf klebende Folie übertragen. Davon wurde dann das Negativ in einem Kreis ausgeschnitten, um dann den Rest an den entsprechenden Ort zu kleben. Jetzt konnte mit der Farbe grosszügig über das ganze Logo gestrichen werden. Als diese dann getrocknet war, musste bloss noch die Klebefolie abgezogen werden, um das Logo zu offenbaren.

Abb. 22: Das Ausgeschnittene Logo

Abb. 23: Die geklebte Folie vor dem Auftragen der Farbe

Für das Banner wurde ein ähnliches Verfahren gewählt. Das Logo wurde am Computer entworfen und auf eine schwarze Klebfolie kopiert. Die Figuren wurden dann ausgeschnitten, während der Rest auf das Plexiglas geklebt wurde. Zuhinterst wurde schliesslich noch eine blaue Folie überzogen, um den Symbolen und dem Schriftzug Farbe zu verleihen. Die Leuchtstoffröhre, welche innerhalb des Kabinettes montiert wurde, leuchtet nun durch die blaue, semitransparente Folie und bringt das Banner zum Leuchten.

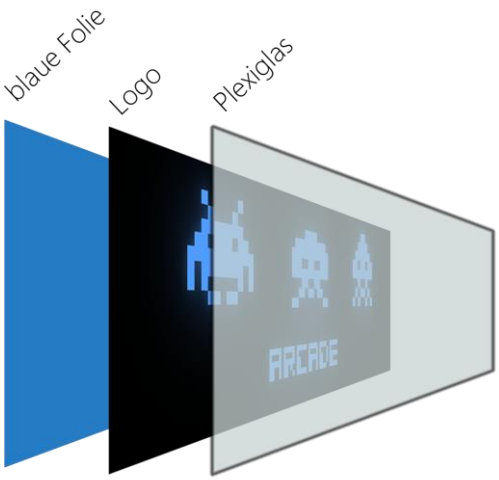


Abb. 24: Aufbau des Banners

Abb. 25: Das fertig montierte Banner

Mit diesem letzten Schritt war das Arcade Cabinet komplett.

Abb. 26: Das fertige Arcade Cabinet

5. Informatik

5.1 Das ArcadeOs – Eine kurze Einführung

Das ArcadeOs ist die zentrale Software, auf der das Arcade Cabinet läuft und für welche die Spiele geschrieben werden. Es ist allerdings nicht ein wirkliches Betriebssystem, wie es der Name suggeriert, sondern bloss eine Java Applikation, welche auf einem gewöhnlichen Windows Computer ausgeführt wird.

Das ArcadeOs hat drei Hauptaufgaben. Die erste ist es, sich mit dem Arduino zu verbinden, Eingabebefehle von den Knöpfen und Joysticks entgegenzunehmen und sie schliesslich zu verarbeiten. Als zweite Aufgabe, muss es eine Grundstruktur für die Spiele, welche dafür geschrieben werden können, schaffen und diese in das System einbetten. Das Ziel dieser Aufgabe ist es, dass man sich beim Programmieren von Spielen voll und ganz auf die eigentliche Sache konzentrieren kann und sich keine Sorgen darüber machen muss, wie diese dann gestartet oder ins Menü integriert werden. Als letzte Aufgabe, soll die Software auch ein Startbildschirm enthalten, aus welchem die Spiele ausgewählt und gestartet werden können.

5.2 Input Handling

Dieses Kapitel beschreibt, wie die Benutzereingaben dem ArcadeOs übermittelt werden und dort die nötigen Events auslösen. Dieser Vorgang ist in zwei Programme aufgeteilt. Das eine wird auf dem Arduino ausgeführt und liest die Inputs ein, das zweite ist Teil des ArcadeOs auf dem Computer und hat die Aufgabe, die Informationen zu entschlüsseln und dem System weiterzuleiten. Dazwischen kommunizieren die beiden Geräte durch eine serielle Schnittstelle. Bei dieser Sorte von Datenübertragung werden Bits nacheinander über eine Leitung durch den USB-Anschluss übertragen.

In einem ersten Schritt soll nun das Programm auf dem Arduino beschrieben werden, dessen Hauptaufgabe es ist, die Eingaben einzulesen. Der Schlüssel dazu ist die Methode „`read_shift_regs()`“, welche einen Integer-Wert, also eine Zahl zurückgibt, welche aus den einzelnen Bits (Nullen und Einsen) besteht, die aus den Eingabegeräten gelesen werden.

Die Methode `read_shift_regs()` beginnt damit, eine parallele Ladung auszulösen, indem es die Pins `clockEnablePin` und `ploadPin` für fünf Mikrosekunden umschaltet. Das heisst, dass die Eingänge der Schieberegister die Ladungen übernehmen, welche von den Knöpfen und Joysticks durchgelassen werden. Sprich: Im Falle eines Knopfdruckes nehmen sie positive Ladung an, ansonsten eine negative.

```
digitalWrite(clockEnablePin, HIGH);
digitalWrite(ploadPin, LOW);
delayMicroseconds(5);
digitalWrite(ploadPin, HIGH);
digitalWrite(clockEnablePin, LOW);
```

Die Schieberegister, welche im vorigen Schritt ihre Ladung aufnahmen, werden nun in einer Schleife eingelesen. Das Bit an der vordersten Stelle der Schieberegister wird in der Variablen `bitVal` gespeichert. Der Integer `bytesVal`, welcher alle Eingaben in sich ansammelt, fügt dann `bitVal` hinzu. Die Schleife schliesst dann ab, indem es den `clockPin` kurz auf „HIGH" stellt. Dadurch wird ein Shift in den Schieberegister ausgelöst und die Bits werden um eine Stelle nach vorne geschoben. Dieser ganze Prozess wiederholt sich dann, bis alle Bits in der Variablen `bytesVal` gespeichert sind.

```
for (int i = 0; i < DATA_WIDTH; i++) {
      bitVal = digitalRead(dataPin);

      bytesVal |= (bitVal << ((DATA_WIDTH-1) - i));

      digitalWrite(clockPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(clockPin, LOW);
}
```

Sind die Inputs eingelesen, bleibt noch die Aufgabe übrig, diese an den Computer zu senden. Dazu muss beim Starten eine serielle Verbindung mit dem Computer hergestellt werden. Auf Seiten des Arduinos ist das sehr simpel und geschieht mit dieser einzigen Codezeile. Die einzige Angabe die man machen muss, ist die Frequenz der Datenübertragung, damit der Computer mit derselben Geschwindigkeit liest, wie der Arduino sendet. Diese wurde hier auf 9600 Herz gestellt, was bedeutet, dass 9600 Bits pro Sekunde verschickt werden.

```
Serial.begin(9600);
```

In die Variable `pinValues` wird dann das Ergebnis aus `read_shift_regs()` gespeichert und mit den letzten gesendeten Daten (`oldPinValues`) verglichen. Unterscheiden sich diese, so werden sie an den Computer gesendet und darauf für den nächsten Vergleich in `oldPinValues` gespeichert. Diese Prozedur wird dann, solange das System läuft, in der Schleife `loop`, mit jeweils 5 Millisekunden Pause zwischen zwei Durchläufen, wiederholt.

```
void loop() {

      pinValues = read_shift_regs();

      if(pinValues != oldPinValues) {
            Serial.println(pinValues);
            oldPinValues = pinValues;
      }

      delay(5)

}
```

Damit ist die Arbeit am Arduino getan, widmen wir uns also dem Code des ArcadeOS. Wir beginnen mit der Methode `initSerialPort()`, welche gleich beim Start des Systems aufgerufen wird und die Verbindung herstellt. Am PC ist diese Aufgabe jedoch nicht gleich leicht mit nur einer Zeile zu bewältigen, weshalb der dafür nötige Code nur grob erklärt wird.

Der Computer beginnt damit, den USB-Anschluss zu finden, an welchen der Arduino angehängt ist. Dazu geht er in einer Schleife alle Ports durch, bis er den richtigen gefunden hat. Darauf wird der serielle Port geöffnet und wie zuvor mit der richtigen Datenrate, sowie anderen Parametern versehen.

Ist eine Verbindung hergestellt, so kann ein `BufferedReader` erstellt werden, welcher eingehende Nachrichten lesen kann. Zum Abschliessen wird noch ein `EventListener` hinzugefügt und dessen Benachrichtigungen aktiviert. Das bewirkt, dass im Falle eines Ereignisses am Arduino, also einem Event, das Programm kurz pausiert wird und die Methode `serialEvent()` aufruft.

```
inputReader = new BufferedReader(
        new InputStreamReader(serialPort.getInputStream())
);

serialPort.addEventListener(this);
serialPort.notifyOnDataAvailable(true);
```

Bei jeder Änderung an den Knöpfen oder Joysticks wird also die Methode `serialEvent()` aufgerufen, welche dann die Nachricht des Arduinos einlesen, korrekt interpretieren und darauf reagieren muss. Um zu verhindern, dass die neuen Tastenbefehle Exceptions auslösen, indem sie beispielsweise in unpassenden Momenten Variablen einer Schleife überschreiben, werden sie in der ArrayList `inputStorage` zwischengespeichert und zu einem späteren Zeitpunkt in der Hauptschleife des ArcadeOs umgesetzt.

Die Methode beginnt damit, den String vom Arduino einzulesen und in der Variablen `inputLine` zu speichern. Dann muss noch `inputStorage` geleert werden, um ihn mit den neuen Inputs zu füllen.

```
String inputLine = inputReader.readLine();
inputStorage.clear();
```

Der eingelesene String ist aber nicht die benötigte Abfolge von Bits, sondern die Zahl, welche diese 16 Bits ergeben. Als erstes muss der String also als Long, sprich als Zahl interpretiert werden, um diese dann zum gewünschten binären String zu konvertieren. Dieser wird dann schliesslich noch in ein Array von einzelnen Buchstaben, in diesem Falle von einzelnen Bits, gespeichert.

In der for-Schleife werden dann die Bits in `binaryChars` eines nach dem anderen zu Integer konvertiert und in den `inputStorage` gespeichert. Dabei wird noch die Reihenfolge gedreht, da die hintersten Stellen der Zahl den vordersten Pins entsprechen. Weil die Nullen vor der ersten Eins beim Umrechnen in eine richtige Zahl verloren gehen, werden sie am Schluss wieder in den `inputStorage` eingefüllt.

```
String binaryString = Long.toBinaryString(Long.parseLong(inputLine));
char[] binaryChars = binaryString.toCharArray();

for (int i=0; i<binaryChars.length; i++){
        inputStorage.add(Integer.parseInt(Character.toString(
        binaryChars[binaryChars.length - (i+1)])));
}

while (inputStorage.size() < 16) {
        inputStorage.add(0);
}

newInput = true;
```

Die Methode schliesst ab, indem sie den booleschen Wert `newInput` auf `true` setzt. Dadurch erkennt das ArcadeOs beim Durchlaufen der Hauptschleife, dass neue Inputs vorhanden sind. Darauf können drei verschiedene Events aufgerufen werden, je nach Zustand des Inputs. Es gibt `AIPressed`, welches solange aufgerufen wird, wie der Knopf gedrückt ist, `AITyped`, welches einmalig eintrifft wenn der Knopf zum ersten Mal `AIPressed` liefert, und schliesslich noch `AIReleased`, was ebenfalls einmalig beim Loslassen des Knopfes aufgerufen wird.

5.3 Aufbau des ArcadeOs

Als nächstes soll der Aufbau des ArcadeOs und dessen Zusammenspiel mit den ArcadeGames betrachtet werden.

Beim Start des ArcadeOs wird als erstes `initSerialPort()` aufgerufen und die Verbindung zum Arduino hergestellt. Ist das erledigt, so fährt es weiter mit dem Erstellen des Fensters.

Das Arcade-System wird komplett auf einem `JFrame` dargestellt. Für den Vollbildschirmmodus werden zuerst mit `setUndecoratet(true)` alle Ränder entfernt, dann wird das Fenster auf die Grösse des Bildschirmes eingestellt und zuletzt in den Vordergrund gesetzt.

```
wind = new JFrame("Arcade");
wind.setUndecorated(true);
wind.setExtendedState(JFrame.MAXIMIZED_BOTH);
wind.setAlwaysOnTop(true);
```

Diese Methode funktioniert bestens, ist strickt genommen jedoch kein richtiger Vollbildschirmmodus, da es einfach ein vergrössertes Fenster ist. Der „richtige" könnte durch die `GraphicsEnvironment` hergestellt werden, welche auch mehr Kontrolle über die Bildschirmeinstellungen bieten würde. Jedoch stellte sich heraus, dass die Performance dadurch stark leidet und die simplere Methode besser zu funktionieren scheint. Um den Mauscursor zu entfernen, wurde er einfach mit einem durchsichtigen Quadrat mit einer Seitenlänge von einem Pixel ersetzt.

Als nächstes wird ein `PaintPanel` erstellt und in das Fenster hinzugefügt. Die innere Klasse `PaintPanel` ist eine Ableitung von `JPanel`. In ihrer `paint(Graphics g)` Methode wird das ganze Fenster gemalt.

```
paintPanel = new PaintPanel();
paintPanel.setSize(wind.getWidth(), wind.getHeight());
wind.add(paintPanel);
```

Alle Spiele, welche vom ArcadeOs zur Verfügung gestellt werden sollen, müssen zur ArrayList `games` hinzugefügt werden. Der ganze Rest bezüglich der Verwaltung dieser Spiele wird vom System übernommen. Für die Darstellung im Menü werden der Name und das Anzeigebild abgerufen. Ist kein Bild vorhanden, so wird selber eines erstellt.

Abb 27: Default-Bild für das Spiel „Pong"

```
games = new ArrayList();
games.add(new MarsMatrix(wind.getWidth(), wind.getHeight()));
games.add(new Pong(wind.getWidth(), wind.getHeight()));
games.add(new Moorhuhn(wind.getWidth(), wind.getHeight()));
```

Sind alle weiteren Variablen initiiert, so tritt das Programm in die Hauptschleife ein, welche bis zum Schluss ausgeführt wird. Es wird jeweils die `runArcade()` Methode ausgeführt und dann einen Aufruf an die `repaint()` Methode gemacht, welche das Fenster neu rendert. Danach wird das Programm kurz durch `Thread.sleep` pausiert. Wie lange diese Pause dauert wird in der `sleep` Variablen so berechnet, dass das Spiel wenn möglich 60 Bilder pro Sekunde darstellt.

```
while (true) {
     long t = System.currentTimeMillis();

     runArcade();
     wind.repaint();

     long sleep = 1000 / targetFPS + t - System.currentTimeMillis();
     if (sleep > 0) {
          try {
               Thread.sleep(sleep);
          } catch (Exception ex) {ex.printStackTrace();}
     }
}
```

Ist ein Spiel am Laufen, so wird in `runArcade()` auch immer die `run()` Methode des entsprechenden Spiels aufgerufen. Dasselbe gilt für `paint(Graphics g)`, für welches das Graphics-Objekt gleich weitergeleitet wird.

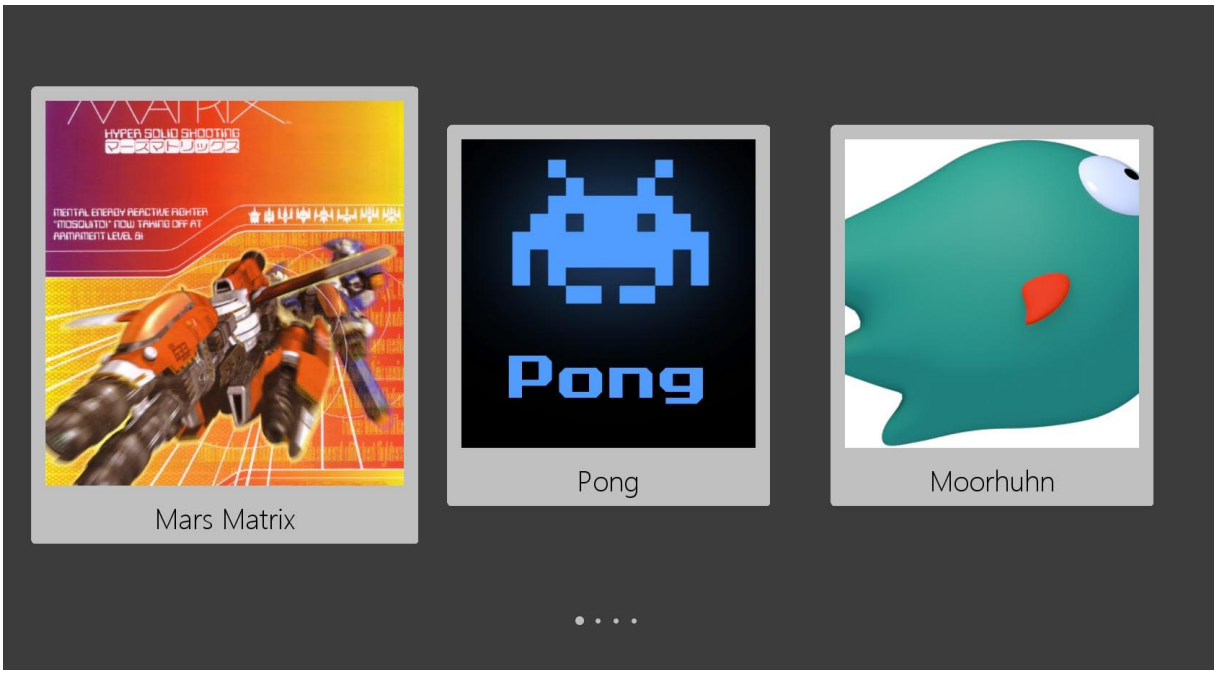


Abb. 28: Screenshot des Menübildschirmes

5.4 Die Klasse ArcadeGame – Struktur der Spiele

Alle Spiele, welche in das ArcadeOs integriert sind, erben von der Klasse ArcadeGame. Dadurch ist die Grundstruktur schon vom System her gegeben und man kann sich vollständig darauf konzentrieren, das Spiel an sich zu programmieren. Wie diese Struktur des ArcadeGames aussieht, soll hier kurz beschrieben werden.

`Init`, `run` und `paint`, sind die drei wichtigsten Methoden der Klasse ArcadeGame und müssen von allen Spielen abgeleitet werden. `Init` wird einmalig beim Starten des Spiels aufgerufen. Dort können alle Variablen initiiert und alle Bilder und Daten geladen werden. Darauf werden 60 Mal pro Sekunde die Methoden `run` und `paint` aufgerufen. In `run` können die jeweiligen neuen Schritte vorgenommen werden, während in `paint` das Graphics-Objekt aus der Hauptklasse gleich überreicht wird, womit das anzuzeigende Bild gerendert werden kann.

```
public abstract void init();
public abstract void run();
public abstract void paint(Graphics g);
```

Die Methoden `close` und `minimize` werden, wie von Namen her ersichtlich ist, aufgerufen, wenn das Spiel geschlossen oder minimiert wird. Sie ermöglichen es, noch einen letzten Schritt vorzunehmen, bevor sie aus der Ausführungsschleife genommen werden, wodurch beispielsweise noch die Spielstände gespeichert werden können. Die Methode `resize` findet auf dem Arcade Cabinet keine wirkliche Verwendung, da dort die Bildschirmgrösse fix ist. Das ganze ArcadeOS ist im Prinzip aber auf jedem Computer spielbar und somit ist `resize` nötig, um die Proportionen bei einer Änderung der Fenstergrösse wieder neu einzustellen.

```
public abstract void close();
public abstract void minimize();
public abstract void resize(int w, int h);
```

Schliesslich muss jedes ArcadeGame noch einige Werte an das System liefern können. Die booleschen Werte `finished` und `minimized` teilen dem ArcadeOs mit, ob das Spiel beendet werden soll oder minimiert wurde. Der String `title` gibt dann den Namen des Spiels zurück und `AppImg` beinhaltet das Bild, welches im Menü angezeigt werden soll. Letzteres ist allerdings nicht obligatorisch, da bei einer Rückgabe eines null-Wertes vom ArcadeOs ein Standardbild erstellt wird.

```
public abstract boolean finished();
public abstract boolean minimized();
public abstract String title();
public abstract Image AppImg();
```

Die einzigen noch zu implementierende Methoden, sind diejenigen zuständig für die Inputs aus dem Arduino und der Tastatur. Diese werden bei einem Event am ArcadeOs direkt an das laufende Spiel weitergeleitet. Mit dieser Liste an Methoden, ist die Grundlage für ein simples Spiel gesetzt und es kann schnell und effizient damit begonnen werden, dieses zu schreiben.

6. Reflexion

Als ich diese Arbeit vor über einem halben Jahr offiziell startete, hatte ich ein grobes, schwammiges Ziel vor Augen, welches sich irgendwo weit weg in der Zukunft befand. Auch wenn ich von Anfang an fest daran glaubte, dass ich dieses Ziel erreichen könnte, so konnte ich mir doch nicht genau vorstellen, welch ein Resultat genau zu erwarten war. Doch diese Zukunft wurde nun zur Gegenwart und ich kann (mit etwas Stolz) behaupten, alles erreicht zu haben, was ich ursprünglich geplant hatte. Vor sechs Monaten bestanden noch viele Unsicherheiten, welche ich damals auszublenden versuchte. Ob der Arduino für das Einlesen der Eingaben geeignet sein würde, ob ich einen Computer und Bildschirm, welche meinen Anforderungen entsprachen, zu einem bezahlbaren Preis finden würde oder wie ich die Bauelemente des Kabinettes zuschneiden sollte, waren beispielsweise Fragen, zu denen ich noch keine Antwort hatte. Dennoch fand ich zu allem eine Lösung. Daran, dass ich für das Zuschneiden der Holzplatten die Hilfe von Girsberger bekommen könnte, hatte ich damals gar nicht gedacht. Auch hatte ich zugegebenermassen etwas Glück, dass ich über Tutti.ch zwei super Offerten für einen PC und Monitor in meiner Nähe entdeckte.

Alles in allem bin ich der Meinung, dass ich das für mich erreichbare Maximum mehr oder weniger erreicht habe. Selbstverständlich bleibt die Qualität auf dem Niveau von eher amateurhafter Handarbeit und gewisse Kleinigkeiten sind noch nicht wirklich perfekt, wie beispielsweise die Schublade, welche nur mässig gut fliesst und manchmal stecken bleibt. Ohne beträchtliche Budgeterhöhung und Aufstieg auf Industrielevel, wären jedoch nur noch kleinere Details zu verbessern und vermutlich keine deutliche Steigerung mehr zu erreichen. Weiter lässt sich darüber streiten, ob es sinnvoll ist, das System so auszulegen, dass ich die Spiele selber schreiben muss, da es zeitlich unmöglich ist, alleine eine gute Auswahl zu programmieren. Weil aber von Anfang an die Idee darauf gründete, dass ich für meine eigenen Games eine Plattform haben möchte, beschloss ich, das so zu belassen. Ausserdem ist hier zu bemerken, dass die Installation von Emulatoren jederzeit und ohne besonders grossen Aufwand noch möglich ist.

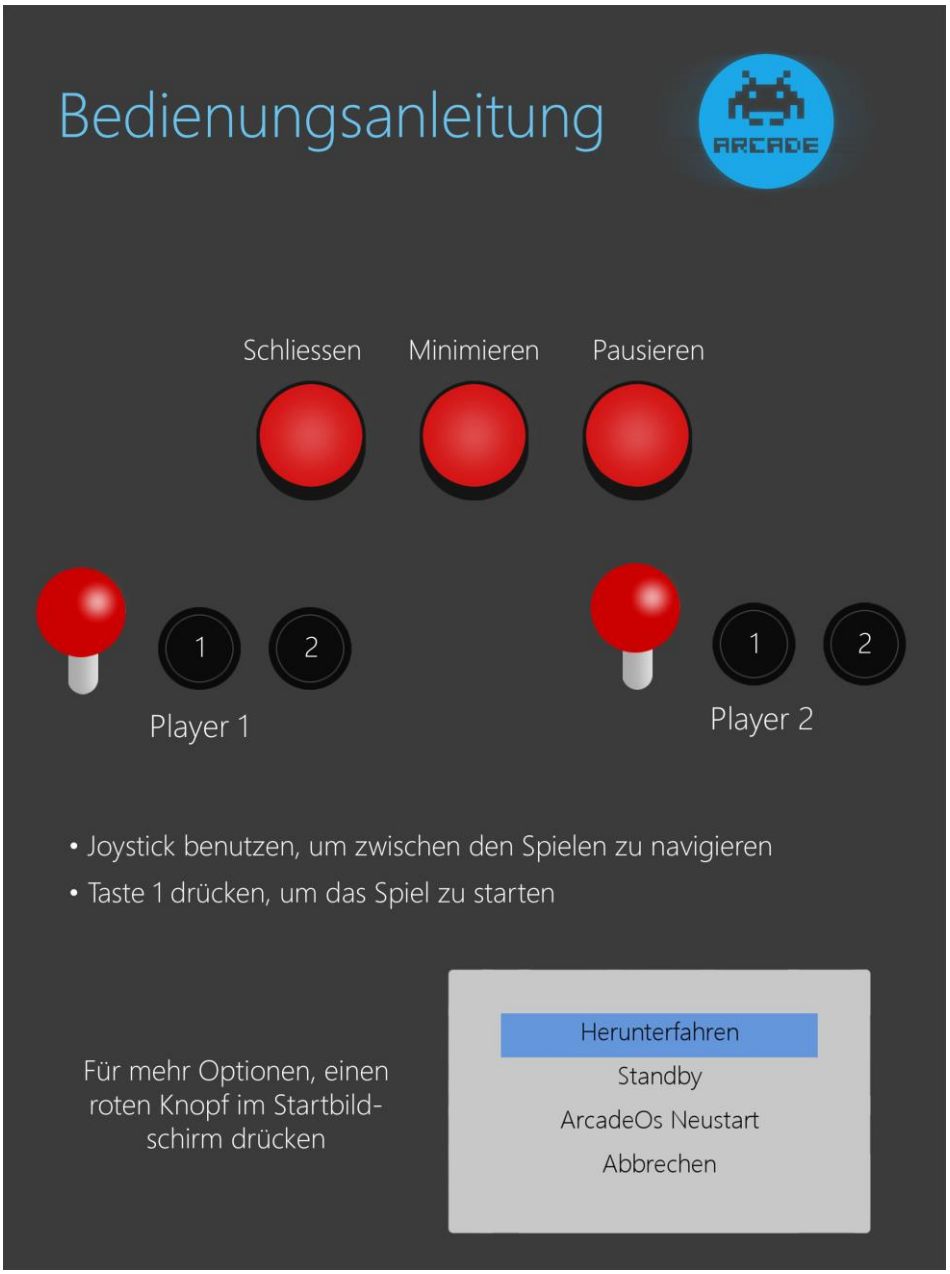
Ich habe mich entschieden, dieses Projekt für meine Maturaarbeit zu wählen, weil ich lieber an einem „richtigen" Produkt als nur an einem Text arbeiten wollte und mir die Vielfältigkeit der Aufgaben gefiel. Ich bin froh, diese Entscheidung getroffen zu haben, denn es war eine Arbeit, für welche ich mich wirklich begeistern konnte und dessen Durchführung mir auch viel Spass bereitete.

Zuletzt möchte ich mich an dieser Stelle noch bei Herrn Forster bedanken, welcher sich bereit erklärte, diese Arbeit mit mir anzupacken und durchzuführen. Weiter geht mein Dank an meine Eltern und meiner Familie, welche mich stets unterstützten und mit Ideen, sowohl hilfreichen wie auch verrückten inspirierten. Und zuletzt noch ein riesiges Dankeschön an Alfred Schaad und der Firma Girsberger für deren grosszügige Unterstützung, ohne welche das Arcade Cabinet wohl kaum die jetzige Form erhalten hätte.

7. Anhang

7.1 Bedienung des ArcadeOs – Eine kurze Anleitung

Die Bedienung des ArcadeOs ist sehr simpel und umfasst nur wenige Befehle. Dennoch sei sie hier der Vollständigkeit halber mit dem folgenden Bild kurz beschrieben.



An der Tastatur übernimmt die „C"- Taste den Schliessknopf, die „M"-Taste den Minimierknopf und die „P"-Taste den Pausierknopf. Ansonsten wird mit den Pfeiltasten navigiert.

An der Tastatur sind noch zwei weitere Befehle möglich, die nicht mit Knöpfen am Arcade Cabinet abgedeckt wurden. Die Escape-Taste schliesst das ArcadeOs und kehrt zu Windows zurück. Die f1-Taste aktiviert die FPS-Anzeige, wodurch im oberen linken Ecken die Anzahl Schlaufendurchgänge pro Sekunde dargestellt wird.

7.2 Quellen

Wikipedia: Arcade-Spiel (9.11.2014) :
http://de.wikipedia.org/wiki/Arcade-Spiel

Wikipedia: Serielle Schnittstelle (9.11.2014) :
http://de.wikipedia.org/wiki/Serielle_Schnittstelle

Arduino: Java (9.11.2014) :
http://playground.arduino.cc/Interfacing/Java

Arduino: SN74HC165N (9.11.2014) :
http://playground.arduino.cc/Code/ShiftRegSN74HC165N

Texas Instruments (9.11.2014) :
https://www.sparkfun.com/datasheets/Components/General/sn74hc165.pdf

7.3 Bilder

Schriftzug Gymnasium Kirchenfeld:
https://intern.gymkirchenfeld.ch/assets/images/GymKirchenfeldBlack.png

Gymnasium Kirchenfeld Logo:
http://www.kinet.ch/gymkirchenfeld/logo_kreator.html

Abb. 7: Foto eines Arduinos:
http://en.wikipedia.org/wiki/Arduino#mediaviewer/File:Arduino_Uno_-_R3.jpg

Abb. 9: Foto des Schieberegisters SN74HC165N:
http://www.oddwires.com/sn74hc165n-texas-instruments-8-bit-parallel-load-shift-registers-2-pack/

Space Invaders Aliens (für das Logo und Defaultbild):
http://www.radio2.be/sites/default/files/images/articles/space_invaders.jpg

Alle weiteren Fotos und Grafiken, sowohl in dieser schriftlichen Arbeit wie auch in der Software „ArcadeOs“, wurden eigens erstellt.