# :Cross Platform Basic

Cross Platform Basic (kurz CPB) ist eine Spieleprogrammiersprache mit der man einfach für eine vielzahl an Systemen Programmieren kann. Hierbei muss man allerdings darauf achten, dass spezielle Dinge nicht auf allen Platformen funktionieren (spezielle Hardware). Außerdem gibt es spezielle Hardware Limitationen, welche man nicht ohne weiteres umgehen kann.

# IDE

CPB wird eine minimalistische GUI mit sich bringen, welche auf Windows, Linux, MacOSX und Pandora läuft. Sie ist selber in CPB geschrieben.

Folgende Features werden enthalten sein:

* Syntax Highlighting
* Hilfe
* Code Übersicht
* Rückgängig/Wiederholen Funktion

# Zielsysteme

Folgende Systeme werden ohne Einschränkungen benutzbar sein:

* Windows
* Linux
* Mac OS X
* Pandora
* GP2X Wiz (fehlende Tastatur, schwache Hardware)

Folgende Systeme werden mit Einschränkung benutzbar sein:

* Nintendo DS (fehlende Tastatur, schwache Hardware)
* Nintendo Wii (fehlende Tastatur)
* Handys (optional)
* Browser
* iPhone/iPod Touch (fehlende Tastatur, optional)
* Android (fehlende Tastatur, optional)
* Sony PSP (optional)

Folgende mögliche Outputs wird es geben:

* C (sehr schnell und auf praktisch jeder Platform verfügbar)
* Java (Um spezielle Platformen zu unterstützen, Android und Handy)
* C# (um es auf der Xbox verwenden zu können)

# Module

CPB ist in verschiedene Modul unterteilt. Ein Modul implementiert zum Beispiel die Möglichkeit Tasten abzufragen oder andere Sachen. Dabei muss man ein Modul entsprechend deklarieren. Jedes Modul muss jede Funktion 2 mal implementieren. Einmal in C Code und einmal in Java Code. Jedes Modul besteht aus drei Dateien.

* XMLCode.xml: Hier wird der XML Code gespeichert, dieser wird dann in die einzelnen Sprachen übersetzt
* JavaCode.j: Hier ist die Java Variante der Funktion
* CCode.c: Hier ist die C Variante der Funktion
* CSharpCode.csharp: C# variante

## Folgende Module wird es geben:

* 2D : Eine auf SDL / Java2D (Java) basierende 2D Engine
* Input
* Sound
* Timer
* Strings
* File
* Network (UDP, TCP)
* Math

Diese Methoden muss jeder Graphik Treiber implementieren:

|  |  |
| --- | --- |
| Funktion: | Was sie macht |
| SetTransform(ScaleX:Float, ScaleY:Float, Rotation:Float) | Setzt die aktuelle Skalierung/Rotation. Funktioniert nur auf Platformen mit OGL /OGLES Unterstützung |
| DrawShape(ShapeID:Int, Poly:Float[]) | Zeichnet eine Form:  ShapeID 0=Rechteck  ShapeID 1=Ellipse  ShapeID 2=Punkt  ShapeID 3=Polygon  (alles relativ zur aktuellen zeichenposition, Poly) |
| DrawImage( Image:TImage) | Zeichnet ein Bild mit aktueller Matrix |
| LoadImage:TImage (Pfad:String, Mask:Int) | Lädt ein Bild Mask=Farbe welche Transparent ist (-1 == Alpha Kanal) |
| Viewport (Width:Int, Height:Int) | Setzt den Sichtbarkeitsbereich, zur Zeichenposition |  |
| SetColor (RGB:Int) | Setzt die aktuelle Zeichen Farbe |
| SetAlpha(Alpha:float) | Setzt den aktuellen Alpha Wert |
| SetBlend(Blend:Int) | Aktueller BlendMode |
| GetImageData:Int[](Image:TImage) | Gibt die Pixel des Bildes im Array zurück |
| Graphics width:int, Height:int, FullScreen:Byte | Erstellt ein Graphik Fenster (Breite Höhe, Vollbild?) |
| Flip | Zeigt den aktuellen Zeichenbuffer, und leert ihn danach |
| SetBlackscreen(RGB:Int) | Setzt die Farbe |
| SetImageData(Data:Int[],Image:TImage) | Setzt ein Bild, welches die Pixel des Arrays hat |
| GrabImageData:Int[](Width:Float,Height:Float) | Nimmt die Bilddaten aus dem Bildschirm, zur aktuellen Zeichenposition |
| SetPosition(X:Float, Y:Float) | Setzt die aktuelle Zeichenposition |
| Reset | Setzt alles zurück |

# Kompiliervorgang

1. Der Code wird in Tokens umgewandelt (Tokenizer)
2. Der Code wird analysiert und es werden die Funktionen, Types und Globale Variablen werden gefunden (Analyser)
3. Der Code wird nach XML umgewandelt (hierbei werden auch Syntax Fehler gefunden). (Parser)
4. Nun werden die Imports eingefügt
5. Nun wird das XML in die Zielsprache übersetzt