RPG Maker XP Kurs
Teil 2

Vorwort

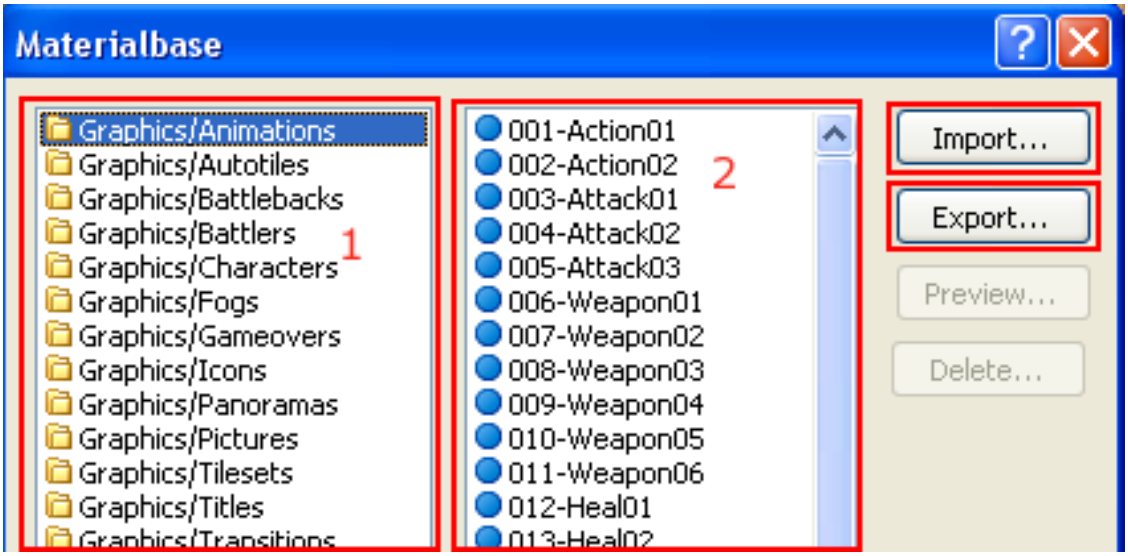
Nun, eigentlich müssten wir genau dort weitermachen, wo wir zuletzt aufgehört haben: Beim Eventscripting. Geplant war eigentlich eine Einweihung in die Benutzung von Move Events, Parallelen Prozessen und AutoStarts zur Erstellung eines Intros. Doch diese Dinge müssen wohl warten. Vielleicht erinnert sich der eine oder andere noch an den ersten Kurs in dem unteranderem die Database erklärt wurde. Zwei wichtige Abschnitte der Database wurden hierbei ausgelassen: Tileset und Animation. Nun, diese zwei Lücken sollen in diesem Kursteil gefüllt werden. Desweiteren wird sich dieser Kurs generell mit dem Einbauen eigener Grafiken in den Maker beschäftigen. Dabei soll jedoch nicht auf die Erstellung von Grafiken eingegangen werden. Da wohl jeder andere Grafikprogramme benutzt und dieses Thema sehr komplex ist, soll es nicht Thema dieses Kurses sein. Stattdessen wird das Einfügen von Grafiken und derren Verwendung Thema dieses Kursteils sein. Dabei ist es vorallem wichtig, dass ihr den ersten Kurs noch einigermaßen in Erinnerung habt, da wir desöfteren darauf zurückgreifen müssen.
Nun denn, dann kann es ja losgehen...

Materialbase

Beim Erstellen eines RPGs können wir uns nicht völlig auf die Standardressourcendatenbank des Makers, genannt ***RTP***, beziehen. In diesem Kurs werden wir auch auf Grafiken zurückgreifen, die nicht in dem *RTP* enthalten sind. Wir müssen sie also erst in unser Projekt einfügen. Dies lässt sich auf zweierlei Arten bewerkstelligen: Über das **Kopieren** oder über das **Importieren**.
Das Kopieren lässt sich kurz und knapp erklären: Sobald ihr den Ordner, in dem sich euer Projekt befindet, öffnet, sollten euch drei Unterordner auffallen. „Audio“, „Graphics“ und „Data“. In dem Unterordner „Audio“ befinden sich die Soundfiles und in „Graphics“ die Grafiken eures Projektes. Sie sind nocheinmal nach ihrem Typ in verschiedene Unterordner eingeteilt. So befinden sich alle *Tilesets* eures Projekts im Unterordner „Tilesets“. Um ein neues *Tileset* eurem Projekt hinzuzufügen müsst ihr also lediglich das gewünschte *Tileset* in diesen Ordner kopieren (rechtsklick auf die Datei, „kopieren“/“copy“ auswählen, den Unterordner „Tilesets“ in eurem Projekt öffnen, dort auf eine leere Fläche wieder mit Rechtsklick draufklicken und „einfügen“/“paste“ wählen).
Sicher fällt euch sofort auf, dass die Unterordner eines neuen Projekts stets leer sind. Es befinden sich keine Ressourcen im Projektordner, obwohl ihr im Spiel doch bereits auf verschiedene Ressourcen zurückgreift. Dies liegt daran, dass der Maker die Dateien aus dem *RTP* nicht in euer Projekt kopiert. Beim Ausführen des Projekts greift das Spiel also immer auf euer *RTP* zurück.

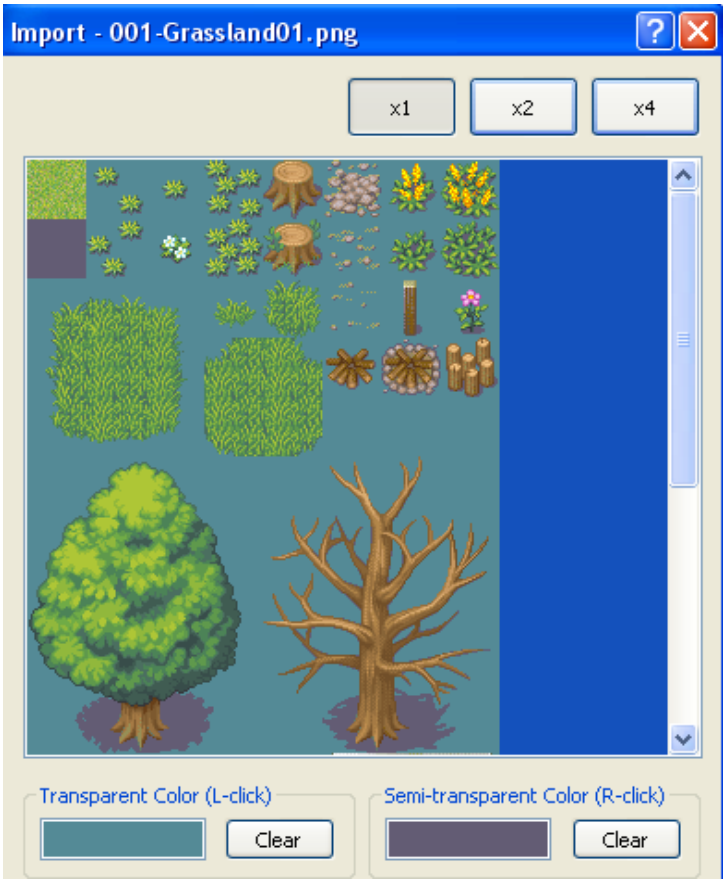
Wenn das Kopieren so einfach geht, weshalb sollte man die Ressourcen dann importieren?
Das Importieren hat verschiedene Vorteile, auf die ich euch gleich aufmerksam machen werde. Doch zuvor besprechen wir kurz, wie das Importieren vonstatten geht.

In der Werkzeugleiste eures Makers befindet sich ein Karteikarten-Symbol. Es führt euch zur **Materialbase**, die ihr ebenso mit der F10-Taste erreichen könnt.



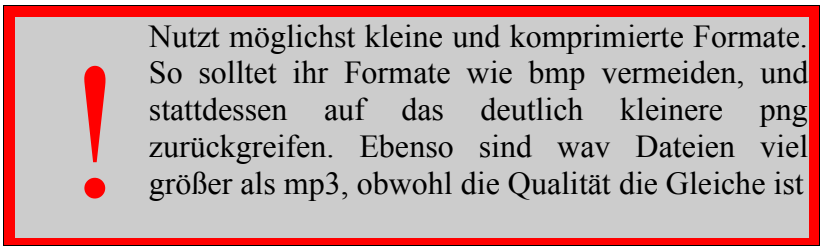
Die Materialbase ist nicht nur eine hervorragende Übersicht über alle Ressourcen, die sich in eurem Projekt befinden, sie ermöglicht es euch auch die Ressourcen zu verwalten. Ihr könnt bestehende Ressourcen löschen, sowie neue Ressourcen einfügen.
In der linken Liste (1) befinden sich alle Unterordner für Sound- und Grafikfiles. Die rechte Liste (2) zeigt an, welche Dateien sich in diesen Unterordner befinden. Diese werden mit einem roten Punkt dargestellt. Es werden jedoch auch alle Dateien angezeigt, die sich in eurem *RTP* befinden. Ihr erkennt sie an dem blauen Punkt.
Mit dem „Delete“ Button könnt ihr eine Ressource löschen. Alle Grafiken könnt ihr über den „Preview“-Button in einem Vorschaufenster begutachten.

Doch der wichtigste Button ist der „Import“-Button.
Über den Import-Button könnt ihr eine Ressource eurem Projekt hinzufügen. Klickt einmal probeweise auf den Import-Button. Es öffnet sich nun ein neues Fenster, in welchem ihr die zu importierende Datei auswählen könnt. Ein Klick auf „Öffnen“ genügt und die Datei wird eurem Projekt hinzugefügt.
Doch die eigentlichen Vorteile des Importierens erfahrt ihr, wenn ihr Grafiken (abgesehen von Battlebacks, GameOver, Titles, Panoramas und Transitions) importiert. Hier öffnet sich nach Auswählen der Grafik nämlich noch ein Vorschaufenster:



In diesem könnt ihr die Transparenz eurer Grafik einstellen.
Zur Auswahl stehen „transparent“ und „semi-transparent“.
Klickt im Vorschaufenster eine Farbe mit der linken Maustaste an. Diese Farbe wird daraufhin *transparent*. Das heißt: sie wird im Spiel nicht mit eingeblendet. Sie ist praktisch unsichtbar. In diesem Beispiel müsste der grüne Hintergrund *transparent sein*, damit man zb. einen Baum auf der *Map* platzieren kann, ohne das der grüne Hintergrund um den Baum herum mit eingeblendet wird.
Semi-transparent heißt soviel wie halbtransparent. Ihr schaltet eine Farbe auf *semi-transparent*, in dem ihr sie mit der rechten Maustaste anklickt. Diese Farbe wird leicht durchscheinend (zu 50% transparent) angezeigt. In diesem Beispiel wäre der graue Schatten, den die Bäume werfen, die Farbe, die semi-transparent geschaltet werden sollte. Wenn wir nun im Spiel einen Baum platzieren, wird der graue Schatten leicht durchscheinend, so dass man den Untergrund (zb. Gras) darunter sieht.
Beachtet dabei, dass ihr immer nur jeweils eine Farbe auf *transparent* und *semi-transparent* stellen könnt. Um das Ganze nochmal zu verdeutlichen zeige ich euch zwei Bilder der gleichen *Map* mit dem gleichen *Tileset*. Einmal wurde die Transparenz richtig durchgeführt (links) und einmal wurde keine Transparenz eingestellt (rechts):

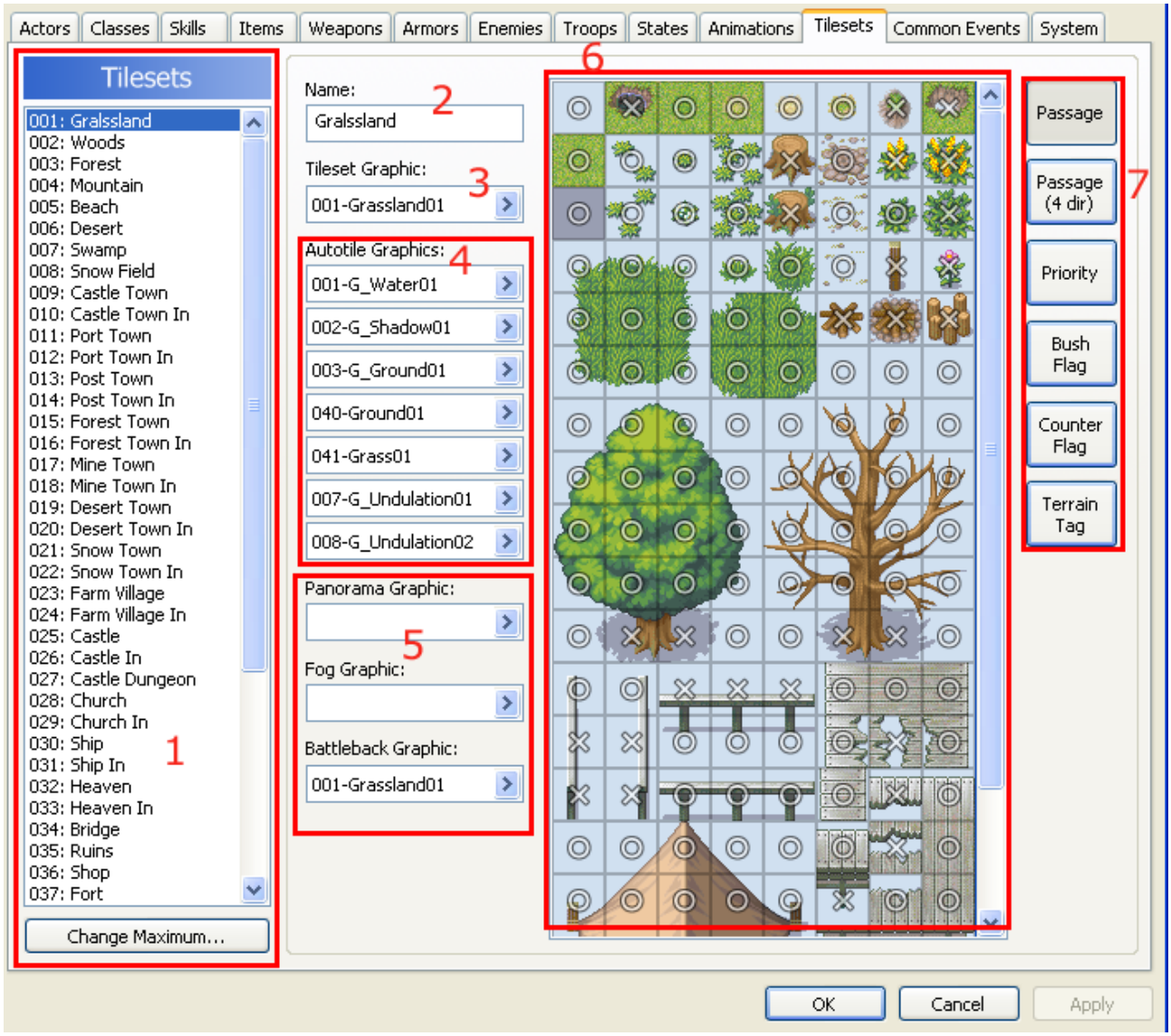
Über ein gutes Grafikprogramm wie Photoshop, Gimp oder PhotoFiltre lässt sich die Transparenz auch manuell einstellen und somit lassen sich sogar verschiedene Transparenzstufen zu nutzen. Wenn ihr derartige Grafikprogramme nutzt, könnt ihr eure Dateien auch direkt in die richtigen Ordner kopieren. Ansonsten ist die Importfunktion immer ein wunderbares Werkzeug für die Transparenzeinstellung. Desweiteren könnt ihr über die Importfunktion auch nur jene Ressourcenformate importieren, die der Maker unterstützt. So wird sichergestellt, dass ihr nicht versehentlich falsche Grafikformate in euer Projekt einfügt, die im Spiel gar nicht dargestellt werden können.
Die *Exportfunktion* stellt das genaue Gegenteil des *Importierens* dar. Hier werden nicht Grafiken in euer Projekt eingefügt, sondern Grafiken aus eurem Projekt hinauskopiert. Diese Funktion werdet ihr aber im Normalfall nicht gebrauchen.



Grafiktypen

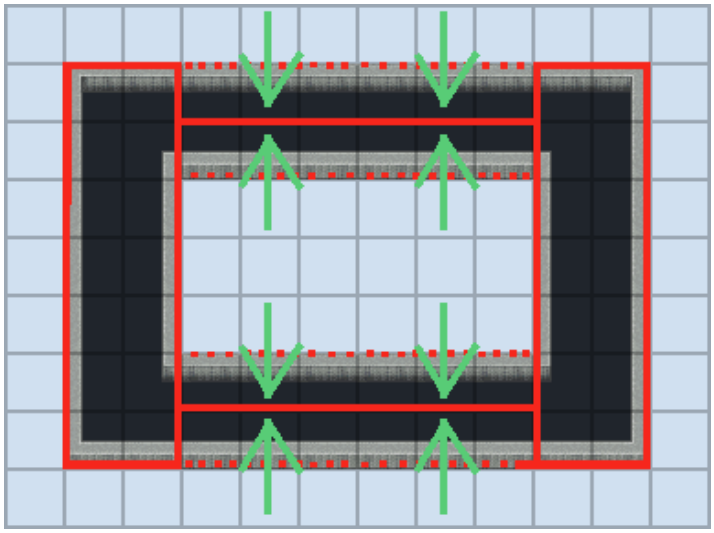
Tilesets

Wie ihr bereits wissen solltet, sind **Tilesets** Grafiken aus denen eure *Map* aufgebaut ist.
Jedes *Tileset* besteht aus 32*32 Pixel großen Quadraten: Den sogenannten Tiles (auf deutsch: Kacheln, bzw. Fliesen). Diese Tiles bilden die Bausteine für eure *Maps*. Das RTP bietet bereits eine Vielzahl an *Tilesets*. Doch werdet ihr euch sicher nicht auf ewig mit den Standardtilesets des RTPs zufrieden geben. Euch steht es frei, neue *Tilesets* in euer Spiel zu importieren, oder bestehende *Tilesets* miteinander zu vermischen. Beim editieren oder erschaffen von *Tilesets* solltet ihr immer darauf achten, dass die Breite der Grafik immer 256 Pixel betragen muss. Nach unten hin hingegen darf das *Tileset* beliebig lang sein. Jedoch muss es immer in 32*32 Pixel große Tiles zerlegt werden können. Das heißt also, die Länge und Breite eures *Tilesets* muss stets durch 32 teilbar sein.
Im Gegensatz zu den meisten anderen Ressourcen genügt es nicht *Tilesets* zu importieren um sie im Spiel zu benutzen. *Tilesets* müssen zusätzlich noch in der *Database* konfiguriert werden.
Wir haben im ersten Kurs bereits über die *Database* gesprochen. Solltet ihr den Aufbau der *Database* nicht mehr so recht in Erinnerung haben, so seht euch nochmal den ersten Kursteil an. Allerdings haben wir dort auch einige Menüpunkte ausgelassen, unteranderem eben *Animations* und *Tilesets*.
Deshalb werden wir diese Wissenslücke in diesem Teil des Kurses erklären.
Öffnet also die *Database* über die F9 Taste oder über den entsprechenden Button in der Werkzeugleiste. Wählt nun den Abschnitt „Tilesets“. Folgender Dialog sollte nun zu erkennen sein:



Die erste Liste (1) sollte euch bereits aus den anderen Abschnitten der *Database* bekannt vorkommen. Hier sind alle *Tilesets* eures Spiels aufgelistet. Wollt ihr ein neues *Tileset* hinzufügen, so müsst ihr die maximale Anzahl der *Tilesets* über den Button „Change Maximum“ erhöhen.
Wir wollen uns das erste *Tileset*, welches aufgrund eines Rechtschreibfehlers den Namen „Gralssland“ trägt, einmal näher ansehen. Klickt also in der Tilesetliste das erste *Tileset* an.
Im Feld „Name:“ (2) könnt ihr den Namen des *Tilesets* ändern. Wir wollen den Rechtschreibfehler zugleich korrigieren, und schreiben daher „Grasland“ in das Eingabefeld.
Bei „Tileset Graphic:“ (3) wird nun ein *Tileset* aus der Materialbase ausgewählt. Beachtet hierbei: Die *Tilesets* in der Materialbase sind lediglich die Grafiken. Damit ihr die *Tilesets* nutzen könnt, müsst ihr diese zuerst in der *Database* konfigurieren. Hierbei könnt ihr aber beispielsweise auch drei *Tilesets* in der *Database* erstellen, die alle ein und dieselbe Tilesetgrafik benutzen.
Desweiteren kann eine *Map* noch mit einem Panorama, einem Fog (Nebel) und einem Battleback (Kampfhintergrund) ausgerüstet werden (5). Was es mit diesen einzelnen Grafiken auf sich hat, erfahrt ihr später.
Rechts befindet sich ein Vorschaufenster eures *Tilesets* (6). Auf jedem *Tile* ist ein Symbol abgebildet. Die Art der Symbole ändert sich, wenn ihr zwischen den verschiedenen Modi (7) hin- und herschaltet.
Der wichtigste Modus ist der „Passage“-Modus (Durchlass-Modus). Hier wird die Begehbarkeit von Tiles eingestellt. Manche Tiles, beispielsweise Wände, Wasser, Gesteinsbrocken, dürfen vom Spieler nicht betreten werden. Solche Felder werden mit einem **X** gekennzeichnet. Die Tiles, die der Spieler betreten darf, tragen ein **O** Symbol. Sobald ihr auf eines der Symbole klickt, verändert sich dieses.

Wollt ihr ein *Tile*, das begehbar ist und somit ein **O** Symbol trägt, unbegehbar machen, so klickt auf das Symbol. Es wechselt daraufhin zu einem **X** Symbol. Das *Tile* ist nun nicht mehr betretbar.
Die ersten sieben Tiles stellen die *Autotiles* dar.
Sie besitzen noch eine dritte „Passage“-Möglichkeit. Das ⍐ Symbol.
Ein auf ⍐ gestelltes *Autotile* ist von oben und unten her genau um ein Feld betretbar. Von links und rechts ist es nicht durchlässig.
Dieser Screenshot soll dies verdeutlichen. Über die roten, durchgehenden Linien kann der Spieler nicht drüberlaufen. Über die gestrichelten hingegen schon. Wie zu sehen ist, kann der Spieler jeweils ein Feld nach unten bzw. ein Feld nach oben durch den *Autotile* hindurchgehen.
Autotiles mit dem ⍐ Symbol eignen sich vorallem für Deckenwände in einem Gebäude, oder als Blätterdach in einem Wald. Nichtsdestotrotz wird dieses Symbol eher selten gebraucht.



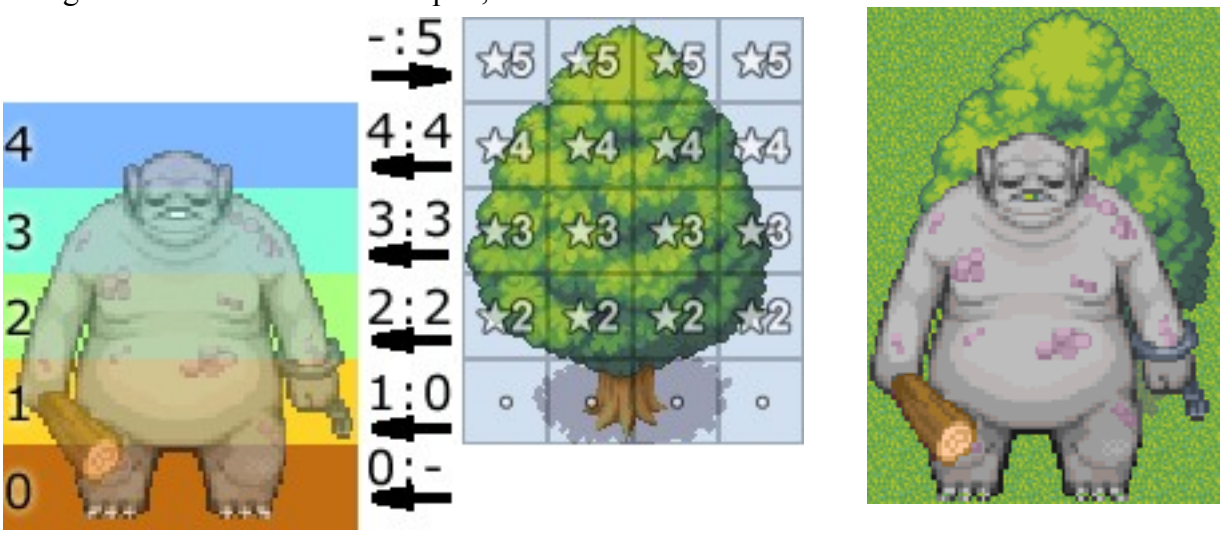
Wenn ihr ein neues *Tileset* erstellt, müsst ihr also in der *Database* jedes *Tile* auf seine Begehbarkeit überprüfen und diese gegebenfalls neu einstellen. Das mag eine aufwendige Arbeit sein, doch auf diese Weise merkt sich der Maker automatisch, welche Felder der Spieler betreten darf und welche nicht.
Der nächste Modus trägt die Bezeichnung „Passage (4 Dir)“ (zu deutsch: 4-Richtungsdurchlauf). Er ist im Grunde genommen nur eine Erweiterung des normalen *Passage*-Modus. In diesem Modus sind auf jedem *Tile* vier Pfeile in alle vier Himmelsrichtung zeigend oder Punkte abgebildet. Ein Pfeil bedeutet, dass das *Tile* von der Himmelsrichtung, in die er zeigt, her betreten werden kann. Ein Punkt dagegen heißt, dass dieses *Tile* von dieser Richtung aus auf unbegehbar gestellt ist.
Habt ihr ein *Tile* im *Passage*-Modus auf **X** gestellt, so ist dieses Feld von allen vier Himmelsrichtungen aus nicht betretbar. In den meisten Fällen reicht dies aus. Doch es gibt auch *Tiles* bei denen etwas Feinarbeit nötig ist.

Dieses Treppentile zum Beispiel darf nur von oben und unten betretbar sein. Schließlich kann man eine Treppe nicht von der Seite aus betreten. In diesem Fall reicht jeweils ein Klick auf die Pfeile, die nach rechts und links zeugen. Statt den Pfeilen werden nun dort zwei Punkte angezeigt. Das *Tile* ist in diesen beiden Richtungen nun nicht mehr durchlässig, kann von unten und oben aber noch betreten werden.
Der nächste Modus nennt sich „Priority“ und ist dafür verantwortlich, dass bestimmte *Tiles* über anderen angezeigt werden. Ihr könnt durch einen Klick auf die Zahlensymbole den Zahlenwert von 0 bis 5 verändern. Standardmäßg hat jedes *Tile* eine Priority von 0. Liegen zwei *Tiles* aufeinander, so wird das *Tile* mit der höheren Priority über dem *Tile* mit der niedrigereren Priority angezeigt. Ist die Priority beider *Tiles* gleich groß, so wird das *Tile*, welches auf der höheren Ebene liegt, über dem anderen *Tile* angezeigt.
Normalerweise sind die Prioritysinstellungen für das Mappen wenig von Bedeutung, da wir durch die Ebenen das Verhältnis der *Tiles* zueinander genausogut bestimmen können.
So wird eine Wiese, die sich auf der ersten Ebene befindet, unter einem Baum auf der zweiten Ebene angezeigt. Es ist also gar nicht nötig, dem Baum eine höhere Priority zu geben.
Bei den *Events* und dem Spieler hingegen verhält sich das ein wenig anders. *Events* befinden sich auf der vierten Ebene: Der Ereignisebene. Sie ist die höchste Ebene, dementsprechend werden *Events* über allen anderen *Tiles* angezeigt.

Unser Held soll aber nicht über einem Baum laufen, sondern unter ihm. Hierfür müssen wir uns der Priority bedienen. Wir müssen also allen *Tiles*, die über dem Spieler und den *Events* angezeigt werden, eine Priority über 0 geben.
Doch zuvor müssen wir uns noch näher mit den Priorityeigenschaften der *Events* beschäftigen. Dort gibt es nämlich einige Besonderheiten.
Events werden auf einem Gitternetz platziert. Das heißt unteranderem auch, dass jedes *Event* nur ein Feld einnehmen kann.
Dieses eine Feld verhält sich genauso wie oben erwähnt: Es wird über allen anderen Ebenen angezeigt, weil es sich selbst in der vierten Ebene, also der Ereignisebene befindet. Jedoch kann die Grafik eines *Events* beliebig groß sein, also auch größer als 32*32 Pixel.
Ein *Event* das 32 Pixel breit und 96 Pixel hoch ist, nimmt von der Grafik her drei Felder auf der *Map* ein. Das unterste Feld der Grafik erhält die Priority 0. Alle folgenden in die Höhe ragenden Felder erhalten jeweils eine um eins höhere Priority.

Wir wollen das am Beispiel eines Riesen verdeutlichen: Der hier abgebildete Riese ist 160 Pixel hoch. Seine Grafik nimmt also fünf Felder/*Tiles* ein. Zum Vergleich hier ein Baum, der ebenfalls fünf Felder hoch ist. Das unterste Feld des Riesen erhält die Priority 0. Jedes höhere Feld erhält eine erhöhe Priority. Warum hat auch der Baum eine aufsteigende Priority bekommen? Wäre es nicht einfacher gewesen, alle *Tiles* des Baumes mit einer Priority von 5 auszustatten? Nun, das Beispiel soll den Sinn des Ganzen erklären. In diesem Fall steht der Riese direkt hinter dem Baum. Die Pfeile sollen das Verhältnis der Priority des Riesen mit der Priority des Baumes verdeutlichen. Der Riese hat immer eine niedrigerere Priority. Außer am Kopf. Daher werden Beine und Oberkörper des Riesen hinter dem Baum angezeigt.

Diesmal steht der Riese direkt vor dem Baum. Nun ist er es, der stets die höhere Priority besitzt und vor dem Baum angezeigt wird.
Das ganze funktioniert auch im Spiel, wie ihr an diesem Screen sehen könnt:



Würde die Baumkrone durchgehend die Priority 5 haben, so würde sie die des Riesen immer schlagen, auch wenn dieser vor dem Baum steht.
Das Einstellen der Priority klingt komplizierter als es ist.
Normalerweise reicht es aus, bei der Priority genauso zu verfahren wie der Maker bei den *Events* automatisch verfährt: Das unterste, auf dem Boden aufliegende *Tile* erhält die Priority 0. Alle darüberliegenden *Tiles* erhalten eine, um 1 erhöhte Priority.

Deshalb bekommt eine solche Säule an ihrem Säulenfuß die Priority 0, an ihrem Schaft die Priority 1 und an ihrem Kapitell die Priority 2.
Außerdem muss eine Priority nur an den *Tiles* eingestellt werden, die durchlässig sind. Denn ein *Tile*, das nicht begehbar ist, kann normalerweise auch nicht über einem *Event* liegen.

Die nächsten Modi sind von geringer Bedeutung und werden nur in Ausnahmefällen gebraucht.
Im „Bush Flag“ Modus steht euch ein Punkt-Symbol, sowie ein Wellen-Symbol zur Auswahl. Jedes *Tile*, welches mit einem Wellensymbol gekennzeichnet ist, bewirkt, dass die untersten 12 Pixel eines *Events*, welches sich auf dem *Tile* befindet, transparent werden.

In diesem Screen läuft der Spieler durch ein Gewässer, welches im „Bush Flag“ Modus ein Wellensymbol erhielt. Bei genauer Betrachtung fällt auf, dass die Füße des Spielercharakters transparent sind.

In diesem zweiten Screen wurde dasselbe Gewässer im „Bush Flag“ Modus mit einem Punkt versehen. Dementsprechend sind die Füße des Chars auch nicht transparent.

Der „Bush Flag“ Modus kann für flache Gewässer oder hohe Gräser benutzt werden. Eben alles, wo die Beine eines *Events* „versinken“ können.
Nun kommen wir auch schon zum vorletzten Modus: Dem „Counter Flag“-Modus. Auch er wird nur in seltenen Fällen gebraucht. Um einem *Tile* einen Counter-Flag zu verpassen, müsst ihr das Punktsymbol des *Tiles* anklicken, was sich daraufhin zu einem Raute-Zeichen verändern sollte. Dem Symbol für einen Counter-Flag.

Tiles, die mit einem Counter-Flag versehen sind, übertragen den Tastendruck eines Spielers auf das benachbarte Feld, falls dort ein Push-Key-*Event* stehen sollte.

Counter-Flags werden vorallem für Ladentische eingesetzt. Ein solcher Tisch trennt den Spieler vom Verkäufer. Wird das *Tile* des Tisches mit einem Counter-Flag versehen, so leitet der Tisch den Tastendruck auf das Verkäuferevent weiter.
Der Tastendruck wird stets in die Richtung weitergeleitet, in die der Held zeigt. Außerdem kann der Tastendruck nur dann weitergeleitet werden, wenn auf dem nächsten Feld ein *Event* mit Push-Key Auslöser steht. Mehrfachweiterleitung ist nicht möglich.
Nun kommen wir endlich zum letzten Modus, dem „Terrain Tag"-Modus. Jedes *Tile* erhält hierbei eine beliebige Zahl von 0 bis 7. Die Zahlen haben normalerweise keine Bedeutung und werden deshalb auch nur selten gebraucht. Ihr könnt aber über die Control Variables Funktion die Terrain-Zahl des *Tiles*, auf dem ein *Event* oder der Spieler sich befindet, auslesen lassen. Somit ist es zum Beispiel möglich, ein Script zu schreiben, welches einen Plätscher-Soundeffekt abspielt, sobald der Spieler sich im flachen Wasser befindet. Hierfür muss nur das Wassertile mit einer individuellen Terrain-Zahl versehen werden (zb. 5).
Nun müsste ein *Event* als paralleler Prozess auf der *Map* erstellt. Der Inhalt des *Events* würde im einfachsten Fall folgendermaßen aussehen:

```
Control Variables [single: 0001:Terrain set Character Player´s Terrain Tag]
Conditional Branch [Variable 0001:Terrain same 5]
        play SE [127-Water02]
        wait [20 Frames]
end branch
```

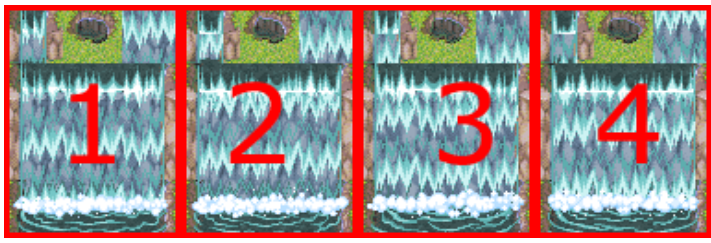
Doch wie bereits gesagt ist auch dieser Modus wenig entscheidend. Wichtig sind eigentlich lediglich der *Passage-Modus* sowie in Einzelfällen der *Priority-Modus*. Auf die anderen Einstellungsmöglichkeiten könnt ihr in den meisten Fällen verzichten.

Sobald ihr alle nötigen Einstellungen vorgenommen habt, klickt auf „Apply" (übernehmen) und schließt die *Database*. Wie ihr eine *Map* mit dem neuen *Tileset* ausstattet, wurde bereits im ersten Kurs besprochen.
Nun können wir also eigene *Tilesets* erstellen oder bestehende *Tilesets* editieren, diese in die *Database* einfügen und letztlich mit ihnen mappen.

Autotiles

Beim konfigurieren eines *Tilesets* in der *Database* mussten wir ja bereits **Autotiles** einstellen. In diesem Kapitel soll geklärt werden, was es mit diesen ominösen Grafiken auf sich hat.
Ein *Autotile* ist ein *Tile*, welches großflächig auf eine *Map* aufgetragen werden kann und dabei einen Rahmen hinterlässt. Ein *Tileset* darf bis zu sieben verschiedene *Autotiles* haben. Beim Mappen bilden die sieben eingestellten *Autotiles* dann die erste oberste Zeile im Tilesetfenster.

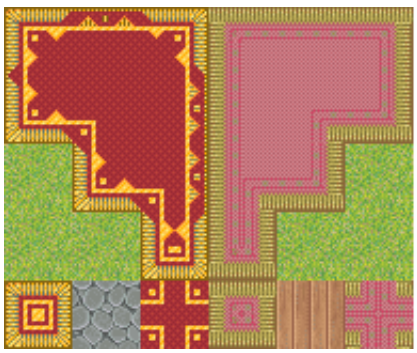
Dieses Wasser hier wäre beispielsweise ein *Autotile*. Aber auch der Rasen, der Teppich etc. Vorallem für den Boden wird oftmals auf *Autotiles* zurückgegriffen. Ein Vorteil von *Autotiles* ist auch, dass diese animiert sein können (wie es beim Wasser der Fall ist). *Autotiles* sind stets 96*128 Pixel groß. Sie können jedoch in der Breite erweitert werden, um so Animationen zu ermöglichen.



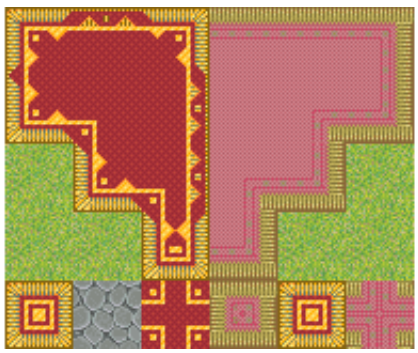
Animiertes Wasserfall-*Autotile*. Die rote Umrahmung samt Zahlen wurde zur Verdeutlichung hinzugefügt. Standardmäßig braucht jedes *Autotile* nur ein 96*128 Pixel großes Kästchen (Feld 1). Wird es in die Länge gezogen, so bekommt es zusätzliche Animationsstufen.

Grundsätzlich ist jedes *Autotile* aus vier Einzelstücken aufgebaut.

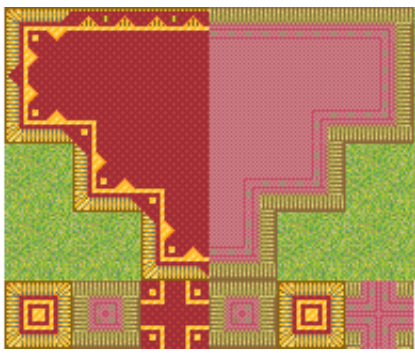
1 Das erste Feld (wir nennen es mal Repräsentationsfeld) dient als Symbol für das *Autotile*. Im *Mapeditor* erscheint statt dem ganzen *Autotile* in der ersten Zeile des *Tilesets* lediglich dieses erste Feld.
2 Dieses zweite Feld (wir nennen es einfacherhalber mal Beziehungsfeld) kennzeichnet die Beziehung des *Autotiles* zu einem anderen *Autotile*. Ist das Repräsentationsfeld eines *Autotiles* gleich dem Beziehungsfeld eines anderen *Autotiles*, so geht das zweite Autotile in das erste Autotile über, sollten beide zusammentreffen. Dieser Screen sollte dieses Phänomen verdeutlichen:



In diesem Screen haben die beiden Autotiles keine Beziehung zueinander. Denn ihre Repräsentationsfelder sind nicht mit den Beziehungsfeldern des jeweils anderen identisch.

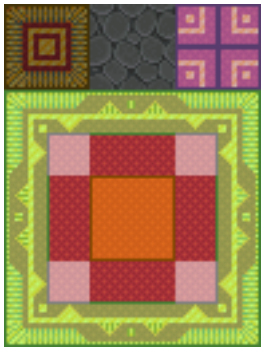


In diesem Screen hat das rechte Autotile eine Beziehung zum linken, da das Beziehungsfeld des rechten Autotiles identisch mit dem Repräsentationsfeld des linken ist. Deshalb geht das linke Autotile in das rechte über, wenn beide nebeneinander in der selben Ebene platziert werden.

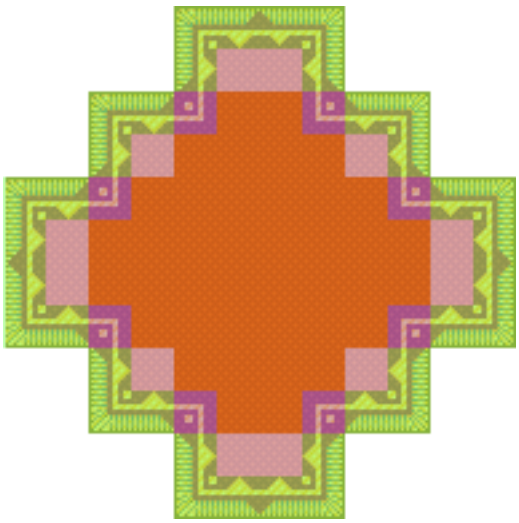


Im dritten Screen haben beide Autotiles eine Beziehung zueinander. Denn das Repräsentationsfeld des rechten Autotiles ist identisch mit dem Beziehungsfeld des linken, und andersherum. Darum gehen beide Autotiles ineinander über, wenn sie nebeneinander in der selben Ebene stehen.

3 Das dritte Feld ist für die Eckpunkte eines Autotiles verantwortlich. Es bestimmt, zusammen mit dem **4**ten Feld den Aufbau des Autotiles.

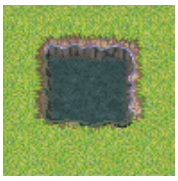


Hier noch einmal unser Teppich-Autotile. Der violett gefärbte Bereich bestimmt die Ecken des Teppichs. Der grün gefärbte Bereich ist der Rahmen. Der orange gefärbte Teil ist der Inhalt des Autotiles.

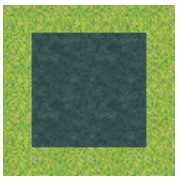


Zum Platzieren von Autotiles auf der *Map* solltet ihr euch gleich einige Tricks merken, die komplizierte Funktionen (wie zb. Die Beziehung zwischen Autotiles) völlig überflüssig machen.

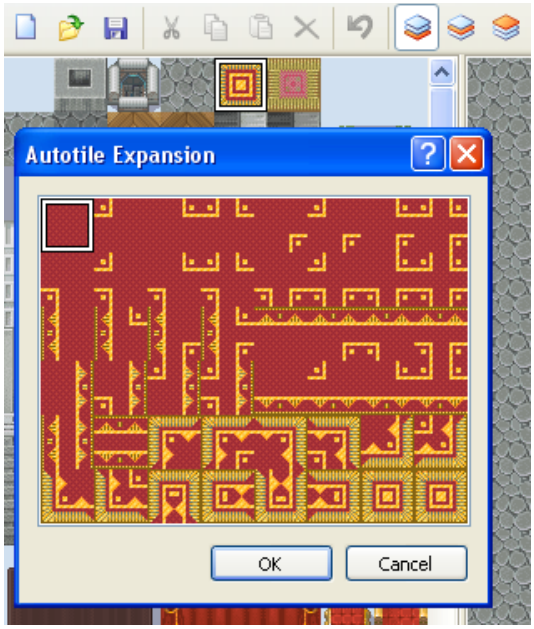
Mit Autotiles mappt man genauso wie mit anderen *Tiles*. Klickt im *Mapeditor* einfach ein Autotile aus der ersten Zeile des *Tilesets* an und füllt eine Fläche auf der *Map* mit diesem Teil. Um euer Autotile herum wird immer der soeben besprochene Rahmen platziert. Wenn ihr also einen kleinen See auf eurer *Map* erstellen wollt, nutzt eines der Wasser-Autotiles die ein Ufer als Rahmen haben (zum Beispiel das Wasser-Autotile, was wir zu Beginn des Kapitels gesehen haben).
Manchmal ist es aber auch sinnvoll, nur mit dem Mittelteil (der im Beispielscreen orange gefärbte) eines Autotiles zu mappen. Wenn wir zum Beispiel einen See ohne Ufer mappen wollen. In diesem Fall haltet, während ihr mit dem Mappingwerkzeug das Autotile auf der *Map* platziert, einfach die Shift-Taste gedrückt.



Hier wurde das Autotile normal aufgetragen



Hier wurde beim Platzieren des Autotiles die Shift-Taste gedrückt gehalten.

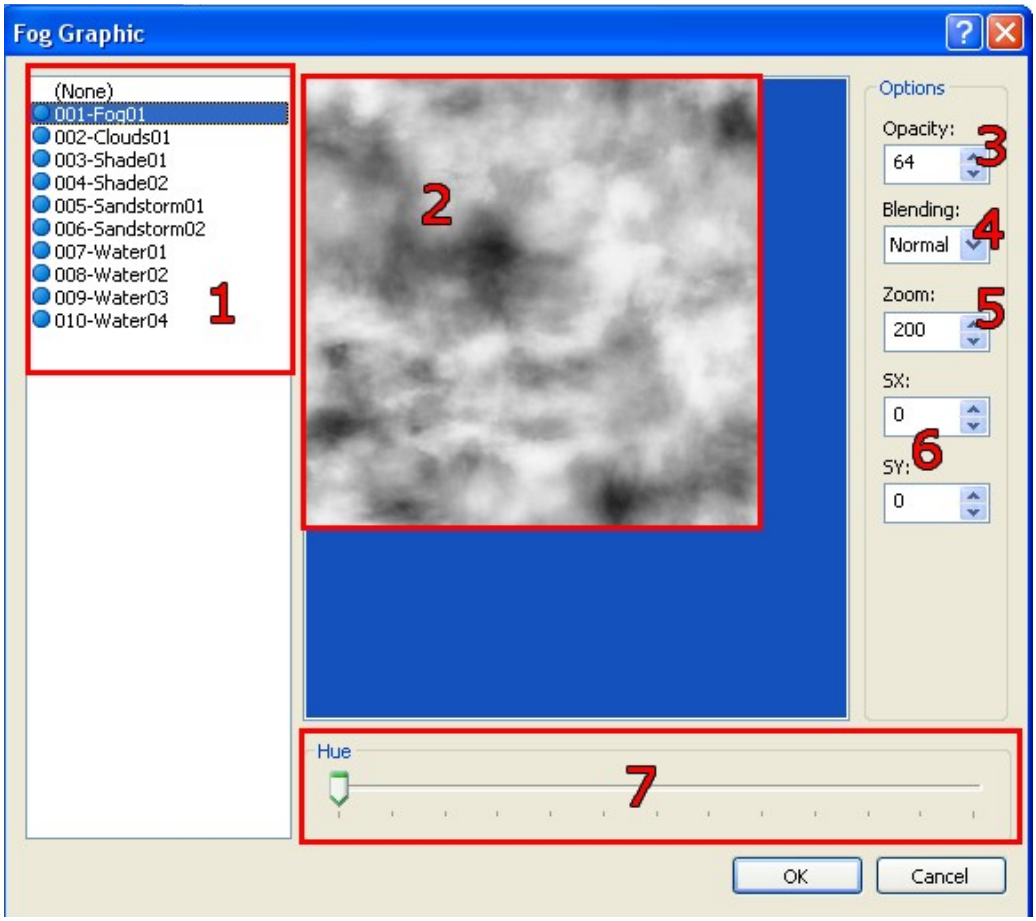


Es gibt jedoch noch eine weiterere nützliche Funktion des Makers beim Arbeiten mit Autotiles.
Wenn ihr im *Mapeditor* ein Autotile in Tileauswahlfenster doppelt anklickt, öffnet sich ein neues Fenster, in dem ihr jedes mögliche *Tile*, dass aus dem Autotile entstehen kann, einzeln auswählen und auftragen könnt.

Ihr seht, Autotiles sind eine geniale, aber auch sehr komplizierte Angelegenheit, mit denen ihr die Qualität eurer *Maps* jedoch stark verbessern könnt. Letztendlich erleichtern Autotiles das Mapping eher, als dass sie es verkomplizieren würden.

Fogs

Wenden wir uns also dem nächsten Grafiktyp zu: Den **Fogs** (zu deutsch: Nebel). Fogs sind eine genial einfache Möglichkeit um einer *Map* eine passende Atmosphäre zu verschaffen.
Im Grunde genommen sind Fogs lediglich Bilder, die über eurer *Map* angezeigt werden. Sie können leicht transparent sein, und sich eventuell sogar bewegen. Damit ermöglichen sie vielerlei Effekte. Zum Beispiel können sie Wolken darstellen, die über eurer *Map* entlangschweben. Oder eine Nebelwand, die einem schaurigen Wald noch mehr Düsternis verschaffen soll. Sie können als Blätterdach verwendet, oder für Licht- und Schatteneffekte genutzt werden. Foggrafiken können beliebig groß sein. Jede *Map* darf nur einen einzigen Fog besitzen. Wollt ihr mehrere Fogs in eine *Map* einblenden, so müsstet ihr ein entsprechendes Script mithilfe von Pictures schreiben. Aber soll jetzt nicht unser Thema sein.
In der *Database* unter *Tileset* könnt ihr jedem *Tileset* ein passenden Fog verpassen.
Wenn ihr doppelt auf das Feld „Fog Graphic:“ klickt, müsste sich folgendes Fenster öffnen:



In der linken Liste (1) könnt ihr, wie gehabt, einen Fog aus eurer Ressourcenliste auswählen, welcher als Vorschaubild (2) rechts daneben angezeigt wird.
Bei „Options“ (3-7) könnt ihr eine Reihe von Einstellungen vornehmen. „Opacity“ (Deckkraft) wird die Transparenz des Nebels eingestellt. 255 ist die höchste Zahl und bedeutet 0% Transparenz. Die Grafik ist also voll sichtbar. In diesem Fall würde das wohl bedeuten, dass der Nebel so dicht ist, dass wir die eigentliche *Map* gar nicht mehr sehn. Deshalb sollte die Opacity runtergeschraubt werden. 0 bedeutet 100%ige Transparenz. Der Nebel wäre dann überhaupt nicht mehr sichtbar. Ein Mittelwert beider Extremen, am besten eher ins niedrigerere tendierend, wäre für einen Nebel ideal. In diesem Fall eben 64. Bei Blending könnt ihr euch zwischen dem „Normal“-, „Add“- und „Sub“-Modus entscheiden. Bei „Add“ und „Sub“ werden dunkle Farben durchlässiger als helle Farben.

Das heißt, dass Weiß, die hellste Farbe, die Opacity erhält, die ihr im Optionspunkt „Opacity“ eingestellt habt, während Schwarz, die dunkelste Farbe, eine Opacity von 0 bekommt und damit unsichtbar wird. Alle anderen Helligkeitsstufen bekommen eine Opacity zwischen beiden Extremen zugewiesen. Bei „Sub“ werden anschließend noch alle hellen Farben dunkel und alle dunklen Farben hell. Im „normal“-Modus erhalten alle Farben des Nebels die ihnen zuvor zugewiesene Opacity -und zwar einheitlich und gleichmäßig. Im Normalfall könnt ihr diesen Modus auf „Normal“ belassen. „Add“ eignet sich eigentlich gut, um schönes Wetter sowie Lichteffekte darzustellen, während „Sub“ sich für Schatteneffekte oder Gewitterwolken bewährt.

Schlechtes Wetter mit Add-Effekt

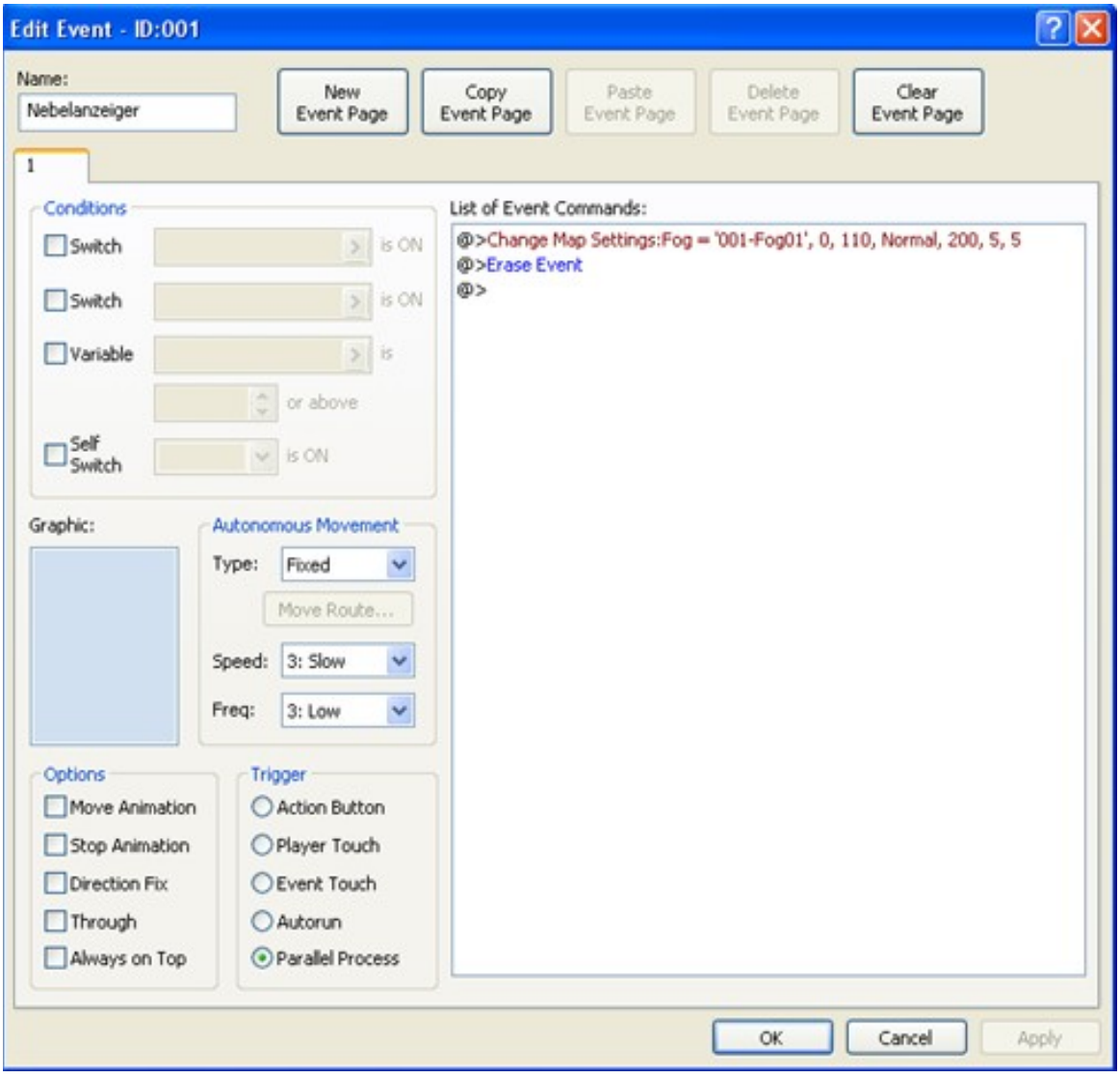
Schönes Wetter mit Sub-Effekt

Die nächste Option heißt „Zoom“ und ist eigentlich schon selbsterklärend. Durch sie kann die Größe der Nebelgrafik in Prozent eingestellt werden. 100 ist die reale Größe. 200 ist die doppelte Größe usw. Der Bildschirm wird übrigens immer voll mit dem Nebel ausgefüllt. Ist die Nebelgrafik kleiner als der Bildschirm, dann wird sie eben mehrmals nebeneinander angezeigt.
Wollen wir mit unserem Nebelpic also viele kleine Wolken darstellen, so verwenden wir einen Zoom von 100. Bei einer Nebelwand dagegen einen Zoom von 800, was das Maximum darstellt.

Die nächsten zwei Optionen „SX:“ und „SY:“ (6) sind für die Bewegung des Fogs zuständig. Ist der Wert auf 0, so bewegt sich der Fog nicht. Dies ist zum Beispiel bei einem Blätterdach im Wald sinnvoll. Wolken und Nebel dagegen sollten sich besser bewegen sachte bewegen. Der Wert neben SX gibt die Bewegung in Richtung der X-Achse an, also von links nach rechts. SY gibt die Bewegung auf der Y-Achse an, also von oben nach unten. Negative Werte bewirken, dass der Nebel sich rückwärts bewegt (ein negativer Wert bei X bewirkt zb. eine Bewegung von rechts nach links). Der Wert durch 8 dividiert ergibt die Anzahl der Pixel, die sich der Fog pro Frame bewegt (pro Sekunde werden ungefähr 40 Frames abgespielt). Ihr könnt einen Wert zwischen -256 und 256 wählen. Habt ihr sowohl bei SX als auch bei SY einen Wert eingetragen, so bewegt sich der Fog diagonal.

Zuletzt könnt ihr mit dem Regler (7) „Hue“ (Farbton) noch den Farbton des Nebels umstellen und somit Flächen, die vorher Rot waren, zum Beispiel Grün machen. Probiert einfach mal den Regler aus, doch nutzt dafür einen farbigen Fog. Bei einem grauton-Fog lassen sich schließlich keine Farben vertauschen.

Nun seid ihr mit den Einstellungen zum Fog auch schon fertig. Wollt ihr aber einen Fog auf einer bestimmten *Map* haben, ohne dass alle anderen *Maps* mit dem gleichen *Tileset* ebenfalls diesen Fog nutzen, so könnt ihr den Fog auch mithilfe eines *Events* einblenden lassen.
Erstellt hierfür ein neues *Event*. Den Trigger schaltet ihr auf „Parallel Process“. Klickt nun doppelt auf die „Event Command Liste“.
Zum Umstellen der Mapeinstellungen benötigen wir den Eventbefehl „Change Map Settings“, den wir auf Seite 2, linke Spalte, vierte Zeile der *Event Commands* finden. Nun können wir, genauso wie in der *Database*, die drei Mapeinstellungen Fog, Panorama und Battleback beliebig umstellen. Ist die Einstellung getätigt, muss das *Event* sich beenden, damit es die Einstellung nicht unendlich oft vornimmt (denn das ist nunmal eine Eigenheit von parallelen Prozessen, die sich unendlich oft wiederholen). Im vorherigen Tutorial haben wir hierfür SelfSwitchs kennengelernt. In diesem Fall eignen sie sich jedoch nicht, denn SelfSwitchs würden das *Event* dauerhaft beenden. Die Einstellungen, die der Eventbefehl „Change Map Settings“ tätigt, gelten aber nur solange, bis der Held die *Map* wieder verlässt. Beim erneuten Betreten der *Map* werden wieder die Standardeinstellungen der *Database* geladen.
Aus diesem Grund benötigen wir einen Eventbefehl, der das *Event* kurzzeitig beendet, es aber beim erneuten Betreten der *Map* wieder startet. Dieser Befehl nennt sich „Erase Event“ und er befindet sich auf der ersten Seite, linke Spalte, 12te Zeile. Damit wäre das *Event* auch schon funktionsbereit.
Hier noch einmal ein Screen, wie der Eventcode aussehen muss:

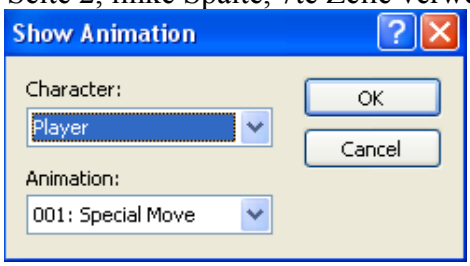


Panoramas, Battlebacks, Titles, GameOvers, Battlers und Icons

Da die nächsten Grafiktypen recht schnell abgearbeitet werden können, bedarf es keiner seperaten Überschriften.
Panoramas sind Bilder im Hintergrund der *Map*. Sie bilden praktisch eine nullte Ebene, denn sie werden hinter der ersten Ebene eingeblendet (ebenso wie Fogs eine fiktive fünfte Ebene bildeten, da sie über allen anderen Ebenen angezeigt wurden). Sie können eine beliebige Größe haben.
Um einer *Map* ein Panorama zuzuweisen, müsst ihr entweder in der *Database* bei den Tileseteinstellungen ein *Tileset* mit einem Panorama ausrüsten, oder ihr verfahrt über den Eventbefehl „Change Map Settings“, wie wir es schon zuvor mit den Fogs gemacht haben.
Damit ein Panorama auf der *Map* angezeigt wird, müsst ihr transparente Untergrundtiles (*Tiles* der ersten Ebene) verwenden.
Wir haben im ersten Kapitel schon darüber gesprochen was passiert, wenn man ein *Tile* der zweiten oder dritten Ebene, welches eine Farbe enthält die in der Materialbase auf transparent gestellt ist, auf der *Map* platziert. Die transparent geschaltete Farbe wird unsichtbar. Stattdessen wird die darunterliegende Ebene gezeigt. Erinnert euch an das Beispiel mit dem Baum.
Um ein Panorama sichtbar zu machen, wählen wir also ein *Tile*, welches die transparente Farbe enthält, aus und platzieren es auf die erste Ebene der *Map*. Im *Mapeditor* wird weiterhin die transparente Farbe gezeigt. Im Spiel dagegen wird entweder das Panorama statt der transparenten Farbe angezeigt, oder, falls kein Panorama eingestellt wird, ein schwarzer Hintergrund.
Das allererste oberste linke *Tile* (welches noch vor den Autotiles kommt) ist komplett in transparenter Farbe gehalten. Wird dieses *Tile* auf der *Map* platziert, so wird das *Tile* komplett durch das Panorama ersetzt.
Das klappt auch bei *Tiles*, die *semi-transparente* Farbe enthalten. Bei denen wird eben die *semi-transparente* Farbe halb durchsichtig über das Panorama eingeblendet.
Das Panorama wird grundsätzlich hinter der *Map* von der obersten linken Ecke aus angezeigt. Ist es kleiner als die *Map*, so wird es eben mehrfach nebeneinander eingeblendet.
Panoramas können genutzt werden, um im Gebirge zum Beispiel einen Ausblick auf die umgebende Landschaft zu bieten. Statt diese selbst zu mappen, setzt man eben ein Panorama ein. Man kann aber auch an anderen Stellen des Spiels Panoramen benutzen. Es gibt sogar Spiele, die Panoramen als Mapersatz benutzen. Ein bekanntes Rm2k-Spiel, welches dieses Prinzip nutzt, ist Sunset over Imdahl.
Der dritte Grafiktyp, welchen man bei *Tilesets* und bei „Change Map Settings“ einstellen kann, ist **Battlebacks**. Battlebacks sind zu Deutsch Kampfhintergründe. Wir hatten im ersten Kurs ja bereits über den Kampfbildschirm gesprochen. Dieser wird in den späteren Kursteilen noch einmal ausführlicher behandelt. *Battlebacks* sollten 640*320 Pixel groß sein.
Ansonsten gibt es zu diesem Grafiktyp nichts weiter zu sagen. Ebenso schnell lassen sich **Titles** (Titelbilder) und **GameOvers** abhandeln. Beide Grafiktypen sollten 640*480 Pixel groß sein. Wenn kleinere Bilder benutzt werden, wird der leer stehende Raum eben mit schwarzem Hintergrund versehen. Welche *Titles* und *GameOvers* das Projekt benutzt, kann in der *Database* unter *System* eingestellt werden (siehe ersten Kurs).
Als letztes behandeln wir in diesem Kapitel noch die **Battlers**. Sie werden während des Kampfes eingeblendet und können sowohl als Helden, wie auch als Gegner verwendet werden. Ihre Größe ist an sich egal, doch damit die gegnerischen *Battlers* noch vollständig zu sehen sind, sollten sie nicht größer als 640*320 Pixel sein. Die *Battlers* von Helden müssen außerdem noch in das kleine Statusfenster des Kampfsystems passen. Daher sollten sie maximal 160*160 Pixel groß sein.
Bei **Icons** verhält es sich ähnlich. Sie werden genutzt im Inventar und in der Skillliste zur Präsentation der gelagerten, bzw. ausgerüsteten Items, sowie der erlernten Skills. Damit sie in das für sie vorgesehene Fenster passen, dürfen sie nicht größer als 24*24 Pixel sein. Da dies schon sehr klein ist, sollte man Icons noch noch kleiner machen.

Animations

Jetzt wird es nochmal richtig komplex und kompliziert. Das letzte Mal in diesem Kursteil, versprochen.
Animations werden in erster Linie während des Kampfes benutzt, um effektvolle Zauber, Zustände, Angriffe oder Items darzustellen. Natürlich können sie genausogut auch an anderen Stellen des Spiels benutzt werden. Hierfür muss lediglich der Event Befehl „Show Animation“ auf Seite 2, linke Spalte, 7te Zeile verwendet werden.

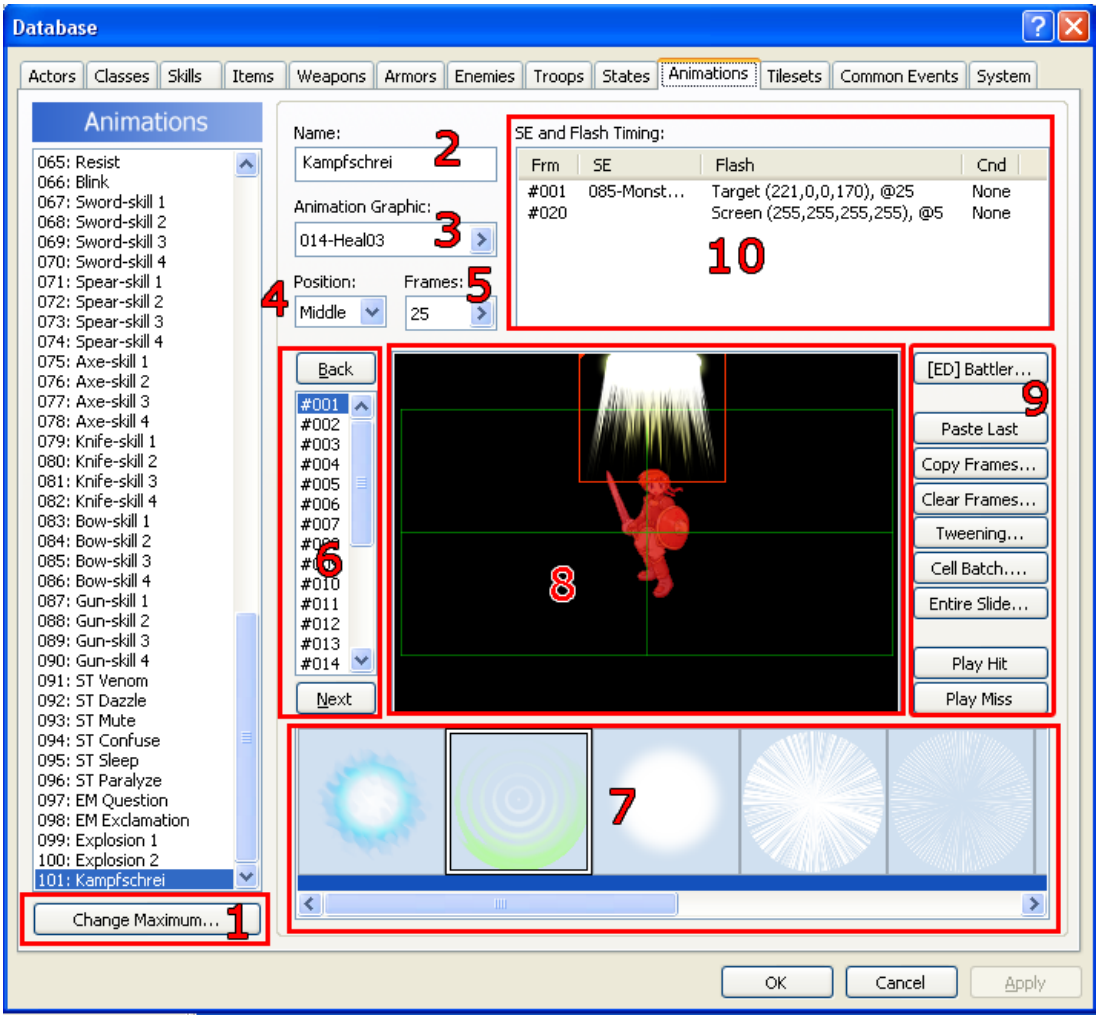


Bei Character wird nun nur noch das *Event* angegeben, über welchem die Animation abgespielt werden soll. Schließlich muss man sich noch in der „Animation:“-Liste die gewünschte Animation heraussuchen.
OK klicken – Fertig!

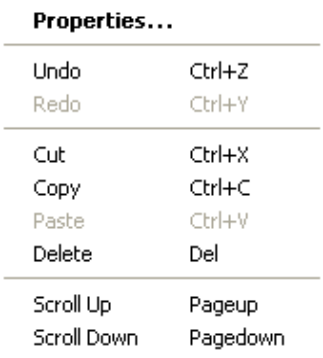
Auch das Integrieren von Animationen im Kampfsystem gestaltet sich einfach. Wir haben ja bereits im ersten Kurs darüber gesprochen. In der *Database* unter *Items*, *Skills*, *Weapons* und *Enemies* befindet sich jeweils eine Option namens „Target Animation:“, „User Animation“ (oder „Attacker Animation“), sowie bei „States“ die Option „Animation“, bei der die gewünschte Animation nur noch aus einer umfangreichen Liste ausgewählt werden muss.
Leider hält diese Liste nicht alles parat, was man für ein gutes RPG braucht. Ihr werdet sicherlich einmal in eine Situation kommen, in der ihr eine eigene Animation einbauen wollt. Nun, das Zeichnen von Animationen sollte kein besonderes Wissen verlangen (außer den Umgang mit Grafikprogrammen). Doch wie auch bei den *Tilesets* ist das Erstellen und Importieren der Grafik nur eine Seite der Medaille. Die andere ist, diese Grafik auch in der *Database* einzustellen.

Doch bevor wir uns dem Fenster in der *Database* samt seinen vielen Einstellungen zuwenden, sollten wir erstmal einige theoretische Aspekte über Animations klären:
Eine Animation-Grafik besteht immer aus mehreren einzelnen Einzelbildern, die wie ein Daumenkino aneinandergereiht sind. Jedes Einzelbild ist 192*192 Pixel groß. Es passen immer fünf Bilder nebeneinander. Aber es können beliebig viele untereinander gereiht werden. Diese Einzelbilder werden **Pattern** (auf deutsch: Model, Muster, Prototyp...) genannt. Doch die Pattern sind nur die Grafiken. Werden sie in einer Animation verwendet, so nennt man sie **Cell** (Zelle) .
In einer Animation können bis zu 16 *Cells* gleichzeitig auftreten.
Eine Animation wiederum ist eine Abfolge von Frames. Um den Vergleich mit dem Daumenkino zu wiederholen: Jede Seite des Daumenkinos ist ein Frame. Die einzelnen Zeichnungen, die in den Seiten vorkommen, sind *Cells*. Diese Vorlage/Schablone, von der diese Zeichnungen abgezeichnet werden, sind die *Pattern*.
40 Frames entsprechen ungefähr 1-2 Sekunden. Viel länger muss eine Animation auch nicht sein.
Nun, soviel zum theoretischen. Jetzt wollen wir sehen, ob die praktische Seite ebenso einfach ist.

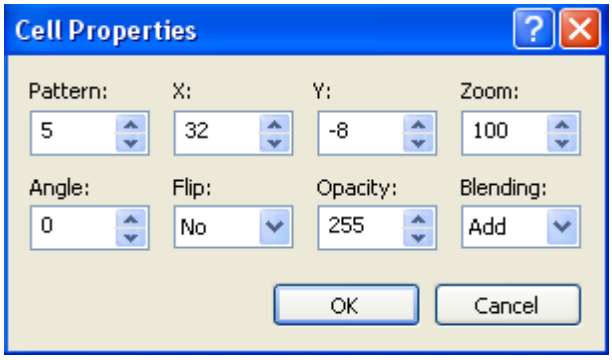
Öffnet also die *Database* und geht in den Abschnitt „Animations“.
Es erscheint nun folgender Dialog:



Links befindet sich wieder die, für die *Database* übliche, Liste mit den verschiedenen Animations. Über „Change Maximum“ (1) könnt ihr die Anzahl der Animationen einstellen. Gebt 101 ein, um eine neue Animation der Liste hinzuzufügen. Wählt diese aus. Bei „Name:“ (2) geben wir der Animation sogleich einen Namen „Kampfschrei“. Womöglich könnte das ja ein neuer Skill für unseren Barbarenhelden werden...
Wählt nun bei „Animation Graphic:“ eine Animation aus der Materialbase aus. Für unseren Skill verwenden wir mal die Grafik „014-Heal-03“.
Als nächstes kommen wir zum Optionspunkt „Position:“. Hier legen wir nun fest, von welchem Standpunkt aus die Animation abgespielt wird. Meistens werden wir zur Position „Middle“ (In der Mitte des Battlers) greifen. Doch möglich wären auch „Bottom“ (Am unteren Punkt des Battlers), Top (Am oberen Punkt des Battlers) oder Screen (in der Mitte des Bildschirms),
Im Optionspunkt „Frames:“ wird eingestellt, wieviele Frames die Animation andauern soll. Wir geben 25 ein. Damit dauert die Animation weniger als eine Sekunde.
Direkt darunter befindet sich eine Liste, in der alle Frames aufgezählt sind. Hier wird der zu bearbeitende Frame festgelegt. Über „Back“(zurück) und „Next“ (nächster) oder (einfacher) über einen Mausklick auf den gewünschten Frame wechseln wir den zu bearbeitenden Frame.
Ganz unten sehen wir nochmal alle *Pattern* (7). Wenn wir einen *Pattern* anklicken und daraufhin auf das mittlere Vorschaufenster (8) klicken, wird ein *Cell* von diesem *Pattern* erstellt. Wie bereits erwähnt können bis zu 16 *Cells* von gleichen oder verschiedenen *Pattern* innerhalb eines Frames erstellt werden. Wenn wir einen *Cell* anklicken, färbt sich dessen Rahmen (der vorher rot war) weiß. Mit gedrückter Maustaste können wir ihn zu einer beliebigen Position verschieben. Klicken wir ihn mit der rechten Maustaste an, so öffnet sich ein kleines Auswahlfenster:



Mit „Scroll Down“ und „Scroll Up“ (Nach unten/oben blättern) könnt ihr die ID der *Cell* verändern. Die erste erstellte *Cell* bekommt die ID 1. Danach erhält jede *Cell* eine um 1 höhere ID. Liegen zwei *Cells* übereinander, so wird die *Cell* mit der höheren ID über der anderen *Cell* angezeigt.
Delete, Paste, Copy, Cut und Undo sind relativ selbsterklärend. Über „Priorities“ öffnet ihr das Eigenschaftsfenster der *Cell*. Ihr könnt dieses auch mit einem Doppelklick auf eine *Cell* öffnen.



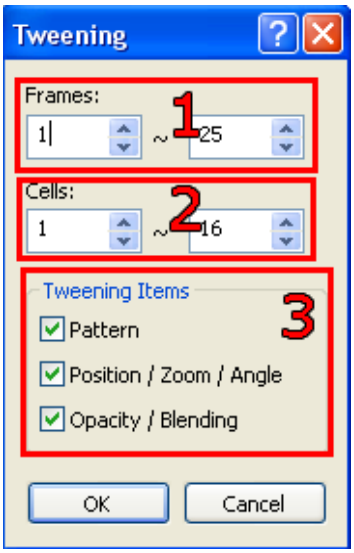
Gehen wir mal die verschiedenen Einstellungen durch. Bei „Pattern“ ist festgelegt, aus welchem *Pattern* die *Cell* gemacht ist. Ändern wir die Nummer des *Pattern*, so verändert sich auch die Grafik der *Cell*. „X“ und „Y“ sind die Koordinaten der *Cell*. Wenn Präzisionsarbeit gefragt ist, könnt ihr hier genaue Werte eingeben, statt die *Cell* mit der Maus zu verschieben. „Zoom“ legt, genauso wie bei den Fogs, die Größe des *Cells* in Prozent fest. Mit „Angle“ (Winkel) könnt ihr die *Cell* entgegen des Uhrzeigersinns drehen. „Flip“ ermöglicht das horizontale Spiegeln der *Cell*. „Opacity“ und „Blending“ erfüllen die gleiche Aufgabe wie bei den Fogs. Nur dass gerade letzteres hier erstaunliche Dienste leistet.

Zurück zum Animationsfenster. Wir können nun also *Cells* platzieren und einige ihrer Effekte ausnutzen. Theoretisch haben wir nun alles um eine Animation zu gestalten. Doch würden wir jetzt für jeden Frame die *Cells* platzieren, so wäre das mindestens genauso viel Arbeit wie ein eigenes Daumenkino zu zeichnen. Das muss doch irgendwie einfacher gehen! Und es geht einfacher. Denn der Maker bietet einige nützliche Optionen (9) die uns eine Menge mühsamer Arbeit ersparen.

Bei „[ED] Battler“ wird der Battler im Vorschaufenster eingestellt. Mit dieser Option könnt ihr die Wirkung eurer Kampfanimation an verschiedenen Objekten vergleichen. Ihr könnt auch ein einzelnen Charakter aus einem Characterset ausschneiden und als Battler importieren, um so herauszufinden, wie die Animation auf ein *Event* wirken würde.
Diese Option ist jedoch nur zu Anschauungszwecken und hat im Spiel selbst keinerlei Einfluss.
Mit „Paste Last“ kopiert ihr die *Cells* des vorherigen Frame in den momentanen Frame. Diese Funktion ist nützlich, wenn ihr von einem Frame zum anderen nur minimale Veränderungen vornehmen wollt. Nach betätigen dieser Funktion müsst ihr nur noch die einzelnen *Cells* verschieben, oder einzelne Funktionen dieser *Cells* ändern, anstatt jede *Cell* mühsam neu zu platzieren.
Wenn ihr eine bestehende Animation mehrmals ablaufen lassen wollt, könnt ihr die „Copy Frame“ Funktion benutzen. zb. Ihr habt eine Animation einer Explosion und wollt, das diese Explosion gleich zwei Mal hintereinander abläuft. Beim Anklicken dieser Option öffnet sich ein neuer Dialog, in dem ihr festlegen könnt, welche Frames kopiert werden sollen (der erste Wert legt fest, von welchem Frame an kopiert und der zweite Wert, bis zu welchem kopiert werden soll), und ab wohin sie kopiert werden (der dritte Wert gibt an, wohin der erste zu kopierende Frame kopiert werden soll. Die weiteren Frames werden nacheinander nach diesem ersten eingefügt).
Wenn also eine Explosion fünf Frames (1,2,3,4,5) einnimmt, und gleich danach erneut ablaufen soll, gebt ihr die Werte 1 ~ 5 to 6 ein.
Mit „Clear Frames“ könnt ihr eine Abfolge von Frames löschen. Wollt ihr zb. eine bestehende

Animation löschen, so klickt „Clear Frames“ an und gebt den Wert 1 und den maximal möglichen Framewert ein.

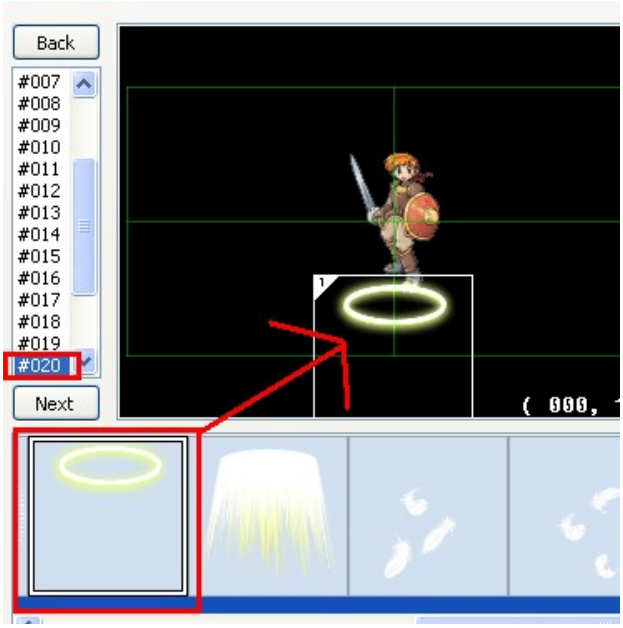
Die nächste Option nennt sich „Tweening“ und ist wohl die genialste Funktion bei der Erstellung von Animations. Sie vervollständigt eine Animation, das heißt: sie füllt Lücken zwischen zwei Frames.
Nehmen wir einmal an, wir wollen einen *Cell* von links nach Rechts bewegen lassen. Diese Bewegung soll 20 Frames andauern. Nun müssten wir im ersten Frame die *Cell* platzieren und fortan bei jeden Frame die Funktion „Past Last“ verwenden und den *Cell* ein paar Pixel weiter nach rechts verschieben.
Die *Tweening* Funktion nimmt uns diese Arbeit ab. Wir müssen nur noch Start- und Endpunkt der Animation setzen. Also im ersten Frame die *Cell* links platzieren und im 20ten Frame dieselbe *Cell* rechts platzieren. Nun betätigen wir die *Tweening* Funktion. Diese vervollständigt die Animation, sie führt von Frame zu Frame eine Bewegung des *Cells* nach rechts aus, bis dieser den Endpunkt erreicht. Durch die *Tweening* Funktion müssen wir also immer nur das Ergebnis einer Animation festlegen. Die einzelnen Schritte werden automatisch erzeugt.
Sehen wir uns diese fantastische Funktion also einmal näher an:



Bei „Frames:“ (1) wird der Start- und Endpunkt des Verfahrens eingestellt. Alle Frames die zwischen diesen beiden Punkten liegen, werden so gefüllt, das ein Übergang zwischen den beiden eingegebenen Frames entsteht.
„Cells“ legt fest, welche *Cells* von diesem Verfahren betroffen sind. Wenn der Übergang nur für einige bestimmte *Cells* gedacht ist, müssen die hier eingestellt werden. Die Werte werden immer als Bereich interpretiert. Also von Wert 1 bis Wert 2. Wenn die Veränderung zwei völlig verschiedene *Cells* (zb. 1 und 8) betrifft, muss das *Tweening* eben zwei Mal hintereinander angewandt werden. Damit nur ein *Cell* dieser Prozedur unterläuft, muss zwei Mal derselbe Wert eingegeben werden.
Bei „Tweening Items“ (3) wird eingestellt, welche Optionen von dem Übergang betroffen sind. Wird nur Opacity/Blending angehakt, so wird nur bei dieser Option ein Übergang erzeugt, während alle anderen Optionen nach einem Standardwert bearbeitet werden.

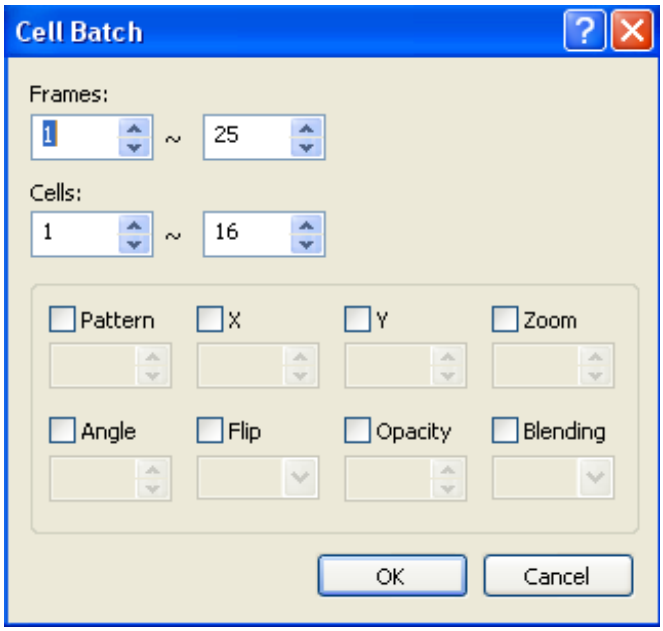
Da das *Tweening* beim Erstellen von Animationen eine enorm wichtige Funktion, will ich es noch einmal an einem Beispiel erläutern:

Wir wählen zuerst den Frame Nummer 1 aus. Nun platzieren wir weit oben eine *Cell*, vom ringförmigen, einem Heiligenschein ähnlichen *Pattern*. Dies soll der Startpunkt der Animation sein.



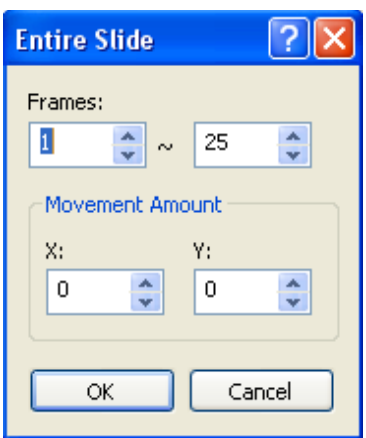
Nun wechseln wir auf den 20ten Frame und setzen erneut denselben *Pattern*, diesmal allerdings weit unten. Dies ist der Endpunkt der Animation.
Als nächstes klicken wir auf „Tweening“ und wählen im anschließenden Dialog die Frames 1 bis 20, sowie die *Cell* 1 bis 1 aus. Bei „Tweening Items“ lassen wir alles angehakt. Klickt nun auf okay und überprüft nun die Frames zwischen 1 und 20. Dort müsste nun eine fließende Bewegung der *Cell* von oben nach unten ablaufen.
Ihr könnt natürlich auch die Opacity, den Angle- oder den Zoomwert in den Übergang mit einbinden. Das macht die *Tweening* Funktion zu einem mächtigen Werkzeug.

Doch was bietet der Maker sonst noch für Funktionen? „Cell Batch“ (Zellstapel) ermöglicht das kollektive Verändern von *Cells*. Ihr somit zum Beispiel die Transparenz aller *Cells* auf allen Frames um einen bestimmten Wert reduzieren.
Beim Benutzen der *Cell Batch* Funktion sollte sich folgendes Fenster öffnen:



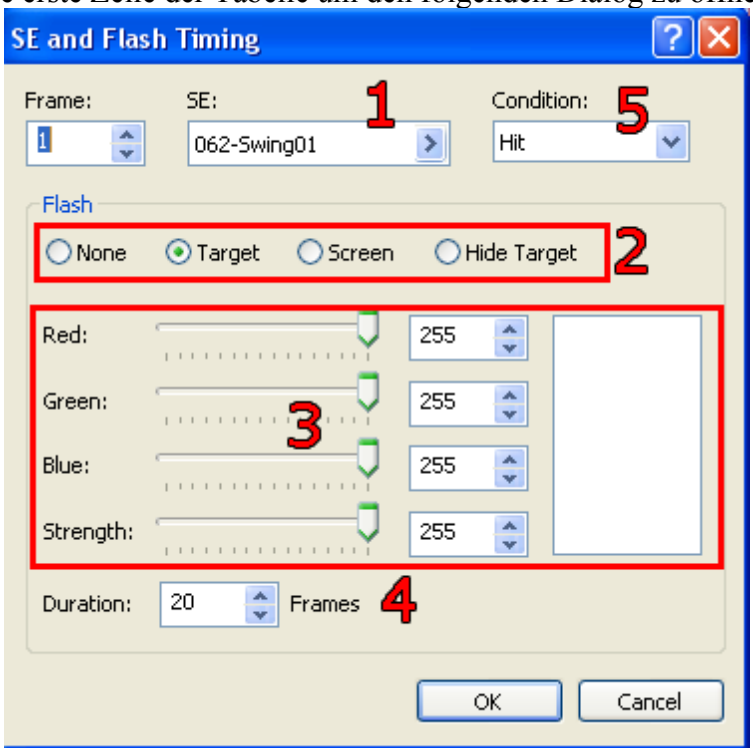
Bei „Frames“ wird wieder eingestellt, von wann bis wann die Funktion wirken soll. Bei „Cells“ wird der Bereich der betroffenen *Cells* genannt, die verändert werden sollen. Danach kommt wieder eine Aufführung der einzelnen Einstellungen, die man bei einem *Cell* vernehmen kann. Sie wirken genauso wie bei den *Cell Properties*. Nur das wir mit der *Cell Batch* Funktion in der Lage sind, mehrere *Cells* gleichzeitig zu verändern, womit wir eine Menge Zeit sparen können. Um eine bestimmte Einstellung zu verändern, müsst ihr nur das gewünschte Element anhaken und den gewünschten Wert eintragen.

Die nächste Funktion „Entire Slide“ (vollständige Verschiebung) funktioniert ähnlich. Sie ermöglicht es, alle *Cells* gleichzeitig in eine beliebige Richtung zu verschieben.



Bei „Frames“ wird wieder die Zeitspanne angegeben, von wann bis wann der Effekt wirken soll. Da der Effekt auf ALLE *Cells* wirkt, kann kein Bereich der *Cells* angegeben werden. Stattdessen muss nur noch die Richtung in X- und Y-Richtung angegeben werden. Denkt

daran: X geht von links nach rechts, Y von oben nach unten! Mit negativen Werten tauscht man diese Richtungen um. Wird sowohl ein X-, als auch ein Y-Wert zugewiesen, so bewegen sich die *Cells* diagonal.
Bevor wir uns den letzten beiden Buttons zuwenden, werfen wir erst einmal einen Blick auf das obige Fenster, beschriftet mit „SE and Flash Timing:“ (10). Hier können Soundeffekte in die Animation integriert, sowie Flash-Effekte (Aufblitzeffekte) erzeugt werden.
Klickt doppelt auf die erste Zeile der Tabelle um den folgenden Dialog zu öffnen:



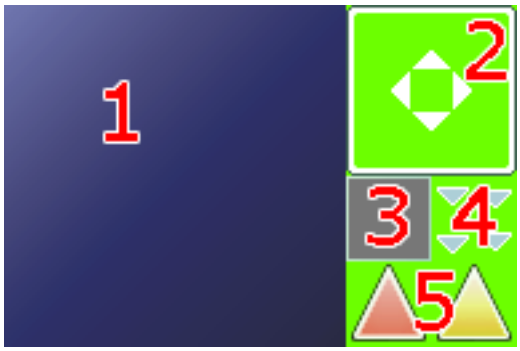
Unter Frame stellen wir ein, ab welchem Frame der Effekt beginnen soll. Soll ein Soundeffekt abgespielt werden, so muss dieser unter „SE:“ (1) ausgewählt werden. Die Punkte 2,3 und 4 müssen nur dann beachtet werden, wenn ein *Flash-Effekt* erzeugt werden soll. Ansonsten wird bei „Flash“ (2) „None“ ausgewählt.
Ein Flash-Effekt ist ein kurzes verfärben eines Objekts in eine andere Farbe. Soll ein Flash-Effekt stattfinden, so muss unter „Flash“ zuerst das Ziel gewählt werden. Mit „Target“ (Ziel) ist der Widersacher im Kampf gemeint. Führt der Held eine Aktion mit dieser Animation aus, so blitzt der Gegner auf. Setzt der Gegner hingegen diese Aktion ein, so ist es der Held, der aufblitzt. Bei „Screen“ (Bildschirm) blitzt/verfräbt sich der ganze Bildschirm. „Hide Target“ (verstecke Ziel) ist eine besondere Option, bei der das Zielobjekt nicht aufblitzt, sondern für kurze Zeit unsichtbar wird. Bei den Farbreglern (3) wird die Färbung des Zielobjektes eingestellt. Mit „Strength“ wird die maximale Deckkraft der Färbung konfiguriert. Sie verhält sich ähnlich wie die Opacity. Ist der Wert auf 255, so färbt sich das Zielobjekt komplett in eine andere Farbe. Ist der Wert nur auf 100, so wird das Zielobjekt nur von einer leicht transparenten, gefärbten Schicht überzogen.
Bei Punkt 4, „Duration“ (Dauer), wird die Dauer des Aufblitzens in Frames angegeben. Der Duration-Wert hält fest, wie lange das Objekt aufblitzen und sich wieder normalisieren soll, oder wie lange es verschwinden muss (im Falle der *Hide Target* Option).
Letztlich gilt es im 5ten Punkt „Condition“ (Bedingung) sich zwischen „Hit“ (getroffen) oder „Miss“ (daneben) zu entscheiden. Hiermit wird differenziert, welcher Effekt beim Erfolg der Aktion, und welcher beim Fehlschlag abgespielt wird. Wenn der Effekt in beiden Situationen eintreten soll, müsst ihr den Effekt eben doppelt, je einmal für beide Optionen, erstellen.
Damit hätten wir auch schon alle nötigen Einstellungen für das Erstellen einer Animation vorgenommen. Mit den letzten beiden Buttons „Play Hit“ und „Play Miss“ könnt ihr die Animation abspielen lassen. Je einmal für den Fall des Erfolgs einer Aktion, und für den Misserfolg.

Animationen sind eine sehr komplizierte und mühevolle Angelegenheit, die man nur mit ausreichender Übung meistern kann. Zum Glück hat der Maker bereits 100 qualitative Animationen voreingestellt.

Characters und Windowskins

Charactersets sind die Grafiken, mit welchen sich der Spieler und *Events* auf der *Map* bewegen. Ein *Characterset* besteht aus mehreren Einzelanimationen, sogenannten *Sprites*. Jedes Set besteht aus 16 *Sprites*. Jeweils vier (horizontal, von links nach rechts) für die Laufanimation und nochmal jeweils vier (vertikal, von oben nach unten) für die verschiedenen Himmelsrichtungen.
Ein *Characterset* kann beliebig groß sein, es muss nur durch 4 teilbar sein. Bedenkt jedoch, dass zwar die Grafik eine beliebige Größe haben kann, doch das *Event* immer nur ein Feld auf der *Map* einnimmt. Ihr könntet also einen Riesen erstellen, der fast die gesammte *Map* einnimmt, doch falls es sich um ein *Push Key Event* handelt, könnt ihr es nur auf dem einen Feld, welches im *Mapeditor* festgelegt wurde, aktivieren. Die Grafik hat also auf die Funktionsweise des *Events* keinen Einfluss.

Windowskins legen einige wichtige Systemgrafiken fest, so zum Beispiel das Aussehen eurer Textbox. Windowskins haben, um richtig funktionieren zu können, eine feste Größe von 192*128 Pixel.



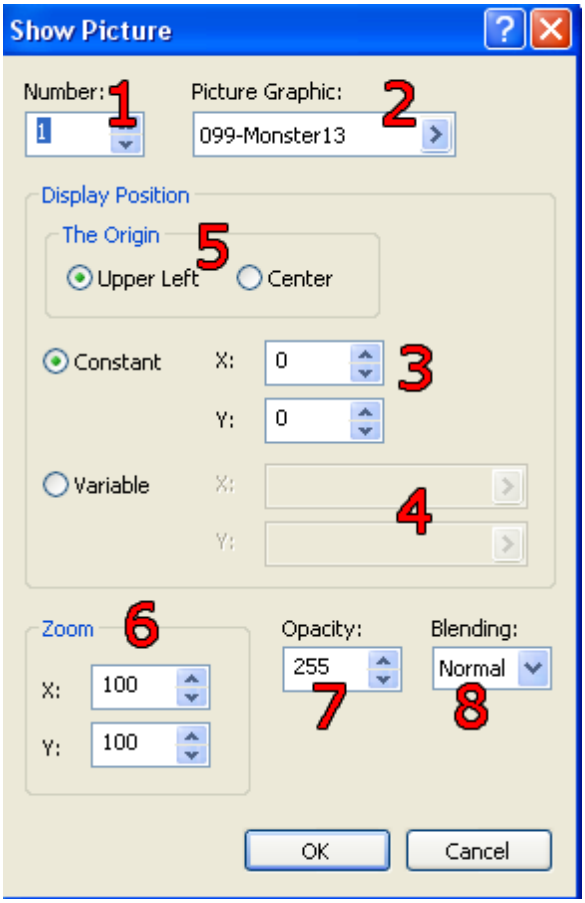
Dieser beschriftete Beispielwindowskin soll den Aufbau eines Windowskins verdeutlichen. Der grüne Hintergrund stellt die transparente Farbe dar. Die große linke Fläche (1) stellt den Hintergrund eines Fensters dar. Mit diesem Hintergrund werden sämtliche Menüfenster, sowie die Textbox gezeichnet. Rechts daneben (2) befindet sich der Rahmen der Textbox, sowie vier Pfeile. Die Pfeile werden dann angezeigt, wenn man innerhalb des Fenster scrollen kann (der Inhalt also größer als das Fenster ist). Darunter befinden sich wiederum zwei Felder. Im linken (3) Feld befindet sich die Grafik, mit dem die Auswahlcursor für die Menüs gezeichnet werden. In diesem Fall wird eine semi-transparente Farbe genutzt, um den Auswahlcursor als verdunkelung zu nutzen. Das Feld daneben (4) enthält die Grafik für den Warte-Button, der kennzeichnet, dass eine Message mit einer Entertaste beendet werden kann. Dieser Button ist in vier Animationsstufen geteilt (jede 16*16 Pixel groß), welche nacheinander ablaufen. Die letzten zwei Felder (5) enthalten die Grafik des Auswahlcursors, mit welchem im Kampfsystem das Ziel einer Aktion ausgewählt wird. Der Cursor ist eine zweistufige Animation und hat eine Größe von 32*32 Pixeln.

Transitions

Transitions (zu deutsch: Übergänge) sind Überblendeffekte, die beim Wechsel zwischen Map und Kampfsystem erscheinen. Normalerweise sind sie 256 farbige/8 bit Schwarz-Weiß Grafiken, mit einer Größe von 640*480. Sollte die Grafik zu klein sein, so wird sie mehrfach nebeneinander angezeigt. Bei einer zu großen Grafik wird nur der passende Teil ausgeschnitten.
Der Übergang zwischen Map und Kampfsystem geht folgendermaßen vonstatten: Dort wo sich beim Transition die dunkelsten Farben befinden, wird zuerst übergeblendet. Bei den weißen Farben findet der Übergang zuletzt statt.
Im Prinzip könnt ihr einfach mit einem guten Grafikprogramm verschiedene Schwarz-Weiß Grafiken entwerfen, diese importieren und ausprobieren. Manchmal sind gerade die Transitions, die durch puren Zufall entstehen, immer noch optisch am schönsten anzusehen.

Pictures

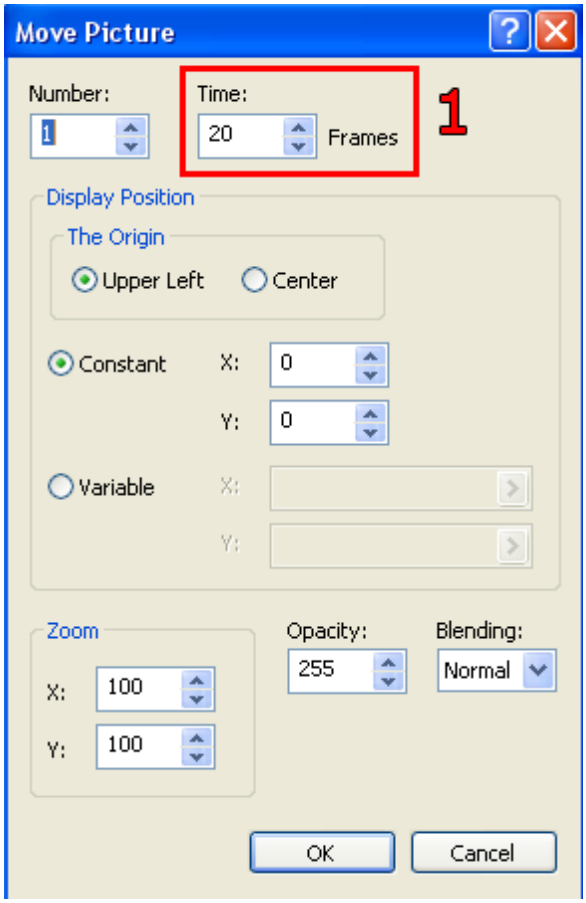
Der letzte und einer der wichtigsten Ressourcentypen: **Pictures**.
Viel kann man zu diesem Thema nicht sagen. *Pictures* können jede Größe haben. Sie unterliegen diesbezüglich also keinen Beschränkungen. In einem standardmäßigen RPG werden *Pictures* nicht gebraucht, daher gibt es auch keine *Pictures* im RTP. Doch beim Scripten sind *Pictures* unerlässlich. Sie lassen sich mithilfe von Eventbefehlen anzeigen, steuern und wieder löschen. Durch *Pictures* lassen sich eigene Menüs oder Kampfsysteme, aber auch einfache Lichteffekte und andere Scripte gestalten. *Pictures* sind im Grunde genommen also einfach nur Grafiken, die für all das benutzt werden können, für die es noch keinen eigenständigen Grafiktyp gibt.
Um *Pictures* anzuzeigen nutzt man den Eventbefehl „Show Picture“, der sich auf der zweiten Seite, rechte Spalte, erste Zeile befindet. Ihr könnt testweise ein *Event* (Auslöser: Push Key) erstellen, welches beim Anklicken ein solches *Picture* anzeigt.
Beim Auswählen des *Show Picture* Befehls sollte sich dieses Fenster öffnen:



Bei „Number:“ (1) wird die ID des *Pictures* festgelegt. Jedes *Picture* hat eine eigene ID. Diese kann eine Zahl von 1 bis 50 sein. Demnach gibt es auch nur maximal 50 *Pictures* gleichzeitig, denn zwei *Pictures* dürfen nicht gleichzeitig dieselbe ID belegen. Wird ein neues *Picture* erstellt, welches die ID eines anderen trägt, so wird das ältere *Picture* gelöscht. Liegen zwei *Pictures* übereinander, so wird das *Picture* mit der höheren ID über dem mit der niedrigeren angezeigt. Bei „Picture Graphic:“ (2) wird die Grafik für das *Picture* ausgewählt (importierst testweise einfach irgendeine Battlergrafik). Die Position, auf der das *Picture* angezeigt wird, muss bei „Constant“ (3) eingegeben werden. Soll die Position des *Pictures* von einer *Variable* bestimmt werden, so muss stattdessen „Variable“ ausgewählt werden. Bei „The Origin“ wird eingestellt, von welchem Punkt aus das *Picture* angezeigt wird. Bei „Upper Left“ wird das *Picture* mit seinem obersten linken Pixel auf die vorher eingegebenen Koordinaten platziert. Wird stattdessen „Center“ ausgewählt, so wird das *Picture* mit seinem zentralsten gelegenen Pixel auf den Koordinaten angezeigt.
Die „Zoom“ Funktion ist ähnlich wie bei den *Fogs*. Nur dass *Pictures* nicht nur gleichseitig vergrößert/verkleinert, sondern auch verzerrt werden können, in dem man bei X und Y unterschiedliche Werte eingibt. „Opacity“ (7) und „Blending“ (8) währendessen funktionieren genauso wie bei den Fogs oder den Animations.

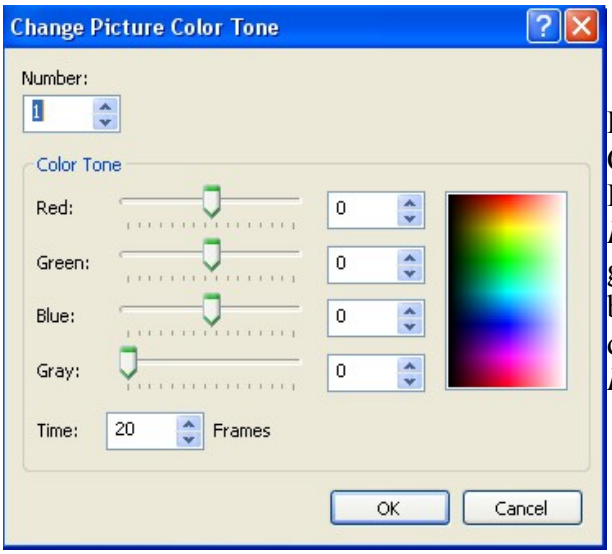
Das besondere an *Pictures* ist, dass sie, nach dem sie platziert wurden, immer noch verändert werden können. Dies geschieht über den Eventbefehl „Move Picture“. Er befindet sich auf der zweiten Seite der *Event Command List*, rechte Spalte, zweite Zeile. Also direkt unter dem „Show

Picture“ Befehl.
Wird dieser Befehl ausgerufen (natürlich muss zuvor ein *Picture* angezeigt werden, damit es verändert werden kann), so öffnet sich folgendes Dialogfenster:



Das Fenster sieht nahezu identisch wie das Fenster der *Show Picture* Funktion aus. Mit einem Unterschied: Statt dem Eingabefeld für die Grafik gibt es nun ein „Time:“-Eingabefeld (1). Tatsächlich bewirkt die *Move Picture* Funktion nur eine Veränderung eines bestehenden *Pictures*. Eine Veränderung der Grafik wiederum ist nur durch erneutes Anzeigen per „Show Picture“ möglich. Stattdessen können hier Einstellungen des *Pictures* wie *Opacity* oder die Koordinaten verändert werden. Das besondere daran ist, dass das *Picture* diese Einstellungen nicht sofort unternimmt, sondern diese in einer fließenden Bewegung ausführt. Ähnlich wie bei der *Tweening* Funktion bei den *Animationen* führt auch hier das *Picture* einen Ablauf bzw. einen Übergang von den vorherigen Einstellungen zu den neuen Einstellungen durch.
Die Dauer für diesen Übergang wird bei „Time“ (1) in Frames festgelegt.

Neben der *Move Picture* Funktion gibt es noch andere Eventbefehle, um den Zustand des *Pictures* zu verändern. So kann man das *Picture* durch die „Rotate Picture“ Funktion rotieren lassen. Dieser Befehl befindet sich direkt unter *Move Picture*. Ihr müsst dabei lediglich die ID des zu rotierenden *Pictures*, sowie dessen Geschwindigkeit eingeben.
Das *Picture* dreht sich dabei immer entgegen des Uhrzeigersinns um den Fixierpunkt, der beim Anzeigen des *Pictures* ausgewählt wurde (entweder oberste linke Ecke oder mitte des Pictures). Bei einem negativen Wert dreht sich das *Picture* im Uhrzeigersinn. Bei 0 hört es auf zu rotieren.



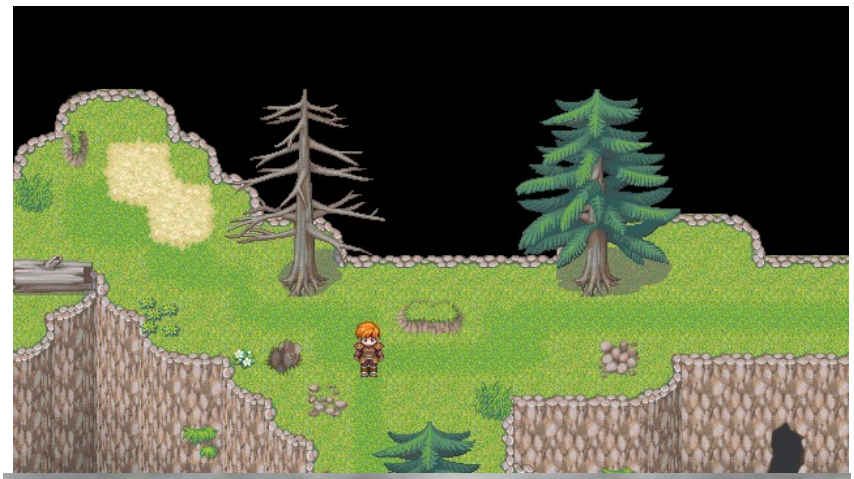
Desweiteren gibt es noch die „Change Picture Color Tone“ Funktion um ein *Picture* umzufärben. Ihr müsst hierzu nur bei Number die ID des *Pictures* eingeben und die Farbregler wie gewünscht stellen. Bei „Time“ wird, genauso wie beim *Move Picture* Befehl, festgelegt, wie lange der Übergang von der vorherigen Farbe des *Pictures* zum neuen Farbton dauern soll.

Damit Pictures, nachdem sie angezeigt wurden, auch wieder entfernt werden können, gibt es den „Erase Picture“ Befehl.
Ihr müsst beim Benutzen des Befehls einfach nur die ID des zu löschenden *Pictures* eingeben um dieses zu entfernen. Ihr könnt übrigens genausogut per *Show Picture* ein leeres *Picture* anzeigen lassen. Dies hat letztlich nahezu denselben Effekt wie eine Löschung des Bildes.

Schlusswort

In diesem Kursteil habe ich das nachgeholt, was ich im ersten Teil versäumt habe. Eine generelle Einweihung in die Arbeit mit Makergrafiken. Nun, vielleicht ist es auch gar nicht falsch, dass dieses eher langweilige und mühsame Thema erst später abgearbeitet wurde und ihr zuvor mit dem interessanteren Themen (und das ist meine ganz subjektive Meinung) wie *Events* und *Database* konfrontiert wurdet. Wer ein gutes Händchen im grafischen Bereich hat, kann nun anfangen sein Projekt mit eigenen Grafiken auszustatten. Doch auch für jene, die sich mit Grafikprogrammen nicht auskennen, sollte es doch recht nützlich sein, über Panoramen und Fogs bescheid zu wissen. Denn dadurch gewinnt so manche öde *Map* doch mächtig an Atmosphäre, wie ich in den beiden unteren Screens demonstrieren möchte.
Im nächsten Kurs werden wir wieder dem zuvor eingeschlagenen Weg folgen und uns weiter dem Eventscripting widmen. Allerdings soll nun auch mehr auf Gamekonzepte eingegangen werden. Das Tutorial soll sich also nicht nur auf die technische/praktische Seite des Makerns beziehen. Wir werden auch eher theoretische Dinge besprechen, die mit dem Maker an sich wenig zu tun haben, aber dennoch essenziell für ein Projekt sind. Praktische Anwendung zu den neuen Themen (vorallem zu den Pictures) wird es wohl erst im nächsten Kurs geben.
Bei Fragen verweise ich, wie immer, auf www.rpga.info
Fragen, Kritik und Vorschläge zum Tutorial können an rmxpkurs@rpga.info gesendet werden!

Hier nochmal die beiden Screens. Einmal ohne und einmal mit zusätzlichen Grafikeffekten.



Register

Befehl	*Seite*	*Spalte*	*Zeile*	*Bedeutung*
Erase Event	1	links	12	Löscht das Event und startet es erst wieder beim erneuten Betreten der Map
Change Map Settings	2	links	4	Verändert die momentanten Einstellungen (Fog, Panorama, Battleback) der Map. Beim Verlassen der Map werden die Einstellungen rückgängig gemacht
Show Animation	2	links	7	Spielt eine Animation auf dem gewählten Event (oder Spieler) ab. Der Eventcode wird während des Abspielens nicht angehalten
Show Picture	2	rechts	1	Zeigt eine Picturegrafik an
Move Picture	2	rechts	2	Verändert den Zustand eines Pictures in einem einstellbaren Übergang
Rotate Picture	2	rechts	3	Konfiguriert die Rotationsgeschwindigkeit und Richtung des Pictures
Change Picture Color Tone	2	rechts	4	Verändert den Farbton eines Pictures in einem einstellbaren Übergang
Erase Picture	2	rechts	5	Löscht ein bestehendes Picture