

■前書き

『吉里吉里 /KAGEX 講座～デジタルノベル製作～』をお手にとって頂きありがとうございます。この本では吉里吉里 /KAGEX を使ったノベルゲームのスクリプトを解説します。昨今のノベルゲームは演出やシステムも高度化しており、スクリプトも難しくなってきています。吉里吉里 /KAGEX ではそれらを比較的簡単に実現する事ができますが、説明書がほとんど無かったり不親切なのが難点です。この状況が本書で少しでも改善できればと思います。

ノベルゲーム製作ツールといえば、昔ながらの吉里吉里や NScripter が有名になっています。最近では他にも YU-RIS や CatSystem2、海外では Ren'Py など、他にも色々と出てきています。この本のターゲットは吉里吉里ですが、市販の解説本などでも取り上げられている KAG の代わりに、KAGEX というシステムを使います。KAGEX は KAG を改造、拡張したもので、今風のノベルゲームを比較的簡単に作ることができます。詳しくは本編に譲ります。KAG と比べても本当に便利になっていて、KAG なら使ったことがある人も驚くかもしれません。

ノベルゲーム製作自体は一昔前と比べれば下火になっているようで寂しい感じです。まだチェックできていないのですが TYPE-MOON さんの新作『魔法使いの夜』が凄いらしいので、また吉里吉里ユーザが増えたりしないかなーと思ったり。しかし吉里吉里をダウンロードしていざ KAG を触ってみると、あまりの落差にあきらめてしまうかもしれません。そこで KAGEX、ついでにこの本なんだぜ。

この場で簡単に筆者の自己紹介もさせてください。自分は Biscrat(ビスクラット) という個人の同人サークルで活動している sakano と申します。まだ大学生の身分ではありますが何本か同人ゲームを作ったりしています。普段は主にノベルゲーム製作のスクリプトを請け負っています。常時お仕事募集中ですので、何かあれば巻末の連絡先までご連絡ください。

泥縄式ながら何本かノベルゲームを作っていました。その経験もできるだけこの本に反映させています。あくまでノベルゲーム製作に関わっている 1 人のプログラマからの視点なので内容にも偏りがあるかと思います。ノベルゲームを完成させるにはスクリプトだけでなくシナリオやグラフィックなど、様々な要素が必要になります。ゲームを作るというのは手間がかかって本当に大変ですが、それなりの楽しさもあつたりします。この本があなたの創作活動の一助となれば幸いです。あなたが作ったゲームを遊べるのを楽しみにしています。

2011 年 12 月 30 日 sakano

Index

1. イントロダクション	5
吉里吉里は色々つくれるフリーソフトウェア	5
TJS は吉里吉里で使うスクリプト言語	5
KAG は吉里吉里でアドベンチャーゲームを作るためのシステム	6
KAGEX は KAG を改造した、ノベルゲームを作るためのシステム	6
スクリプト言語とプログラミング言語	6
アドベンチャーゲームとノベルゲーム	7
「デジタル」ノベルについて	8
デジタルノベル製作の流れ	9
2. 開発環境の準備	13
拡張子の表示	13
吉里吉里 /KAGEX の SDK を入手	13
圧縮 / 解凍	15
素材ファイルの入手	15
開発用フォルダの準備	15
.Net Framework 4 の準備	16
テキストエディタの準備	16
3. スクリプトの基本	18
フォルダ構成	18
テキストの表示	19
吉里吉里の実行について	20
デバッグモードについて	21
ショートカットキーについて	22
画像を表示する	22
伸ばし棒の省略	22
レイヤについて	22
画像の形式について	23
背景の表示	24
画像の位置	26
ピクセル (pixel)	27
立ち絵の表示	28
立ち絵 × オフセット定義	30

表示レベル	31
イベント絵の表示	32
その他の画像の表示	32
コメント	34
省略記法	34
BGM の再生	38
音楽ファイルの形式について	39
効果音の再生	39
true と false	40
属性値 true の省略	41
スクリプトの実行順	41
セーブデータ名の指定	44
セーブデータの消去	45
選択肢	46
空白を含む属性値	46
4. 画像操作スクリプト	48
トランジションの基本	48
その他のトランジション	52
同期 / 非同期トランジションについて	53
不透明度	54
time 属性によるアクション	55
path 属性によるアクション	59
画像の回転	60
回転原点	61
表示原点	63
拡大率	65
反転	66
傾斜率	67
ぼかし	68
ラスタースクロール	70
相対指定	71
パーセント指定	72
初期値指定	73
回転原点のアクション	74
単語登録	74
カメラ操作	75

画面揺らし	81
拡張トランジション	82
トランジションの定義	84
自動トランジションの定義	85
フェード時間の定義	90
バックアップ	90
データのやり取り	91
バージョン管理	91
5. サウンド操作スクリプト	93
同期 / 非同期再生	93
フェードイン / アウト	94
音量指定	95
音量設定について	98
パン	100
ループチューナ	101
ボイス再生	103
遅延実行	107
表情指定のスクリプトとか	108
6. メッセージ操作スクリプト	110
テキストの表示位置	110
色の指定について	112
テキスト枠	112
メッセージレイヤの表示 / 非表示	113
選択肢	115
7. おまけ	119
オブジェクトについて	119
補足	119
シナリオスクリプト / システムスクリプト	119
マクロ	120
envinit.tjs	120
選択肢	121
TJS 的な話	121
変数	121

システムスクリプト	122
リリース	124
メニューとか	124
ボイス	124
動画再生	124
ウェイト	124
[タグ]について	125
ラインモードについて	125
テキスト全画面表示について	125
トランジション	125
時間切れ	125

1. イントロダクション

この本が解説する吉里吉里 /KAGEX とはそもそも何なのか？という話です。どうでもいいといえばどうでもいい話なのでリラックスして読んでもらえればと思います。

■吉里吉里は色々つくれるフリーソフトウェア

吉里吉里は、TJS2 と呼ばれるスクリプト言語を記述することによって、主にマルチメディアタイトルを作成することができるフリー・オープンソースのソフトウェアです。（吉里吉里公式ページより）

よくわからない単語が混じっているかもしれません、要するに「吉里吉里」は色々なことができるフリーソフトウェアです。コンピュータ上でテキストやグラフィックやサウンドなどを出すソフトウェアが作成できます。フリーなので、お金を払わずに使用することができます。

ノベル、シーティング、ロールプレイング、シミュレーション、アクションなど様々なジャンルのゲームやゲーム以外のアプリケーションが大量に製作されています。特に 18 禁エロゲの製作には商業でもかなり使われています。例を挙げると、「TYPE-MOON」の『Fate/stay night』、『Fate/hollow ataraxia』や「Nitroplus」の『STEINS;GATE 8bit』などが吉里吉里を使って製作されたゲームです。

吉里吉里本体は C++ というプログラミング言語を使って作られています。オープンソースとして開発されているので、C++ が使えば本体の改造をする事もできます。ただし、この本は初心者向けとなっているのでこのレベルの話には全く触れることができません。C++ まで扱えるようになると、できることの幅が非常に広がるので興味のある方は挑戦してみてください。

現在の吉里吉里のバージョンは吉里吉里「2」となっていますが、その次の吉里吉里 3 も開発中のようにです。吉里吉里 3 について一番の情報源は吉里吉里制作者 W.Dee 氏の Twitter(https://twitter.com/_w_dee)になります。ただし吉里吉里の話をすることはほとんどありません。吉里吉里 3 では 3D の機能がついたり、Windows 以外でも動作するようになったりする予定だそうです。一応は iPhone や Android などのスマートフォンにも対応するらしいです。あまり開発は進んでいないようですが期待です。

■TJS は吉里吉里で使うスクリプト言語

吉里吉里で何かを作るには「TJS」というスクリプト言語を使う必要があります。TJS は T Java Script の頭字語らしいです。TJS は C++ と比較すれば簡単ではあります

が、プログラム未経験の方が使うには複雑すぎます。また、TJS はほぼ吉里吉里専用の言語のため情報が少なく、自力で勉強する環境が整っているとは言いがたいです。TJS を勉強する際は、JavaScript など TJS 以外の言語でプログラミングの勉強をするをお勧めします。初心者向けの書籍も多くあり、ウェブ上でもたくさん解説されています。プログラミングの知識を得てから TJS の勉強をすればそれほど難しくありません。一見遠回りのようですが最終的には時間の節約になるのではないかと思います。

■KAG は吉里吉里でアドベンチャーゲームを作るためのシステム

吉里吉里を公式サイトからダウンロードしてみると、「KAG」システムが同梱されています。KAG は Kirikiri Adventure Game の略で、アドベンチャーゲーム (Adventure Game) を作るための吉里吉里上で動作するシステムです。

前述の通り TJS を使うにはプログラミングの知識が必要になってしまいます。そこで、プログラミングに興味がない人でもアドベンチャーゲームが作れるように KAG が準備されています。KAG システム自体は吉里吉里で動かすため TJS で作られていますが、KAG を利用してゲームを作るだけなら TJS の知識は必要ありません。非プログラマーにも使える程度に簡単な「KAG スクリプト」だけでアドベンチャーゲームが作れます。

しかし、KAG は何年も昔に作られたシステムのため今時のアドベンチャーゲームを作るには機能が貧弱すぎます。自力で頑張って KAG のカスタマイズをすると大抵の機能をつけることはできますが、そのためにはやはり TJS の知識が必要になってしまいます。ここで KAGEX が登場します。

■KAGEX は KAG を改造した、ノベルゲームを作るためのシステム

「KAGEX」は、KAG を元に渡邊剛氏 (<https://twitter.com/wtnbgo>) が開発した、ノベルゲームを製作するためのシステムです。前述の KAG を拡張する形で作られています。こちらは実際に商業ゲームを開発している会社が開発したシステムなだけあって、最初から様々な機能が利用できます。一般的な形式のノベルゲームであれば「KAGEX スクリプト」を書くことで効率的に製作していくことができます。本書ではこの KAGEX の使い方を紹介していきます。

ウェブや書籍などで吉里吉里 /KAG の解説は見受けられますが、KAGEX の解説はほとんど存在しません。KAG と KAGEX ではかなり違いがあるので、KAG の解説を参考にする際は気をつけてください。

■スクリプト言語とプログラミング言語

細かい話ですが説明しないのも不親切なので書いておきます。現在のところパソコ

ンは自動的に考えて動いてくれるレベルに到達していないので、人間が前もって命令を作つてあげる必要があります。パソコンはその命令に従つてその命令通りに動きまつす。この命令のことを「プログラム」といいます

そしてプログラムを作る作業を「プログラミング」といいます。プログラミングの際には「プログラミング言語」を使います。例えば C++、Java、Lisp、Fortran、Prolog など、大量にあります。「スクリプト言語」はプログラミング言語の一種で、比較的簡単なプログラミング言語のことです。こちらの例としては JavaScript, Lua, Awk など、やっぱり大量にあります。吉里吉里で使用されている TJS もスクリプト言語の一種になります。余談ですが吉里吉里では JavaScript や Squirrel など TJS 以外のスクリプト言語を使うこともできます。もし使いたい、となれば調べてみてもいいかもしれません。

KAGEX スクリプトという KAGEX で使用するスクリプト言語の書き方解説が本書のメインコンテンツになっています。

■アドベンチャーゲームとノベルゲーム

KAG は「アドベンチャーゲーム」、KAGEX は「ノベルゲーム」を作るシステムだと説明しました。アドベンチャーゲームとノベルゲームって何が違うんでしょうか？おそらく広く共有された定義は無いんじゃないかなと思っています。

個人的には『ポートピア連続殺人事件』のようなコマンド入力式ないしコマンド選択式がアドベンチャーゲームです。ノベルゲームといえば、最近の 18 禁ノベルゲームの大部分を占めている、物語の合間にたまに選択肢が挟んであるようなゲームになります。人に聞いてみると後者のゲームをアドベンチャーゲームとも呼ぶようなので私が間違っているようです。

テキストが全画面表示だとアドベンチャー、下半分に表示してるとノベルなどという分類もあるようで、人によってバラバラだなあと思うわけです。はたまた、ノベルなどゲームではないという人もいて、事実何らプレイヤーが介入できず見ているだけの一本道ノベルも存在します。

ゲーム性うんぬんを語るのは荷が重いので割愛します。ゲームを作るのも楽しいですがゲームを研究するのも案外面白いのでちょっと考えてみてはいかがでしょうか。

話を戻して、アドベンチャーゲームの方がノベルゲームよりも自由度が高い、という個人的な感性を捨て切れていないので KAGEX はノベルゲーム用としています。どちらも Kirikiri Adventure Game なのでアドベンチャー用と言ってしまえばいいんで

すけれども。

KAG では用途を限定しきっていないので無理をすれば KAG スクリプトだけで色々できます。それと比べると、KAGEX は特に 18 禁ノベルゲームで一般的な形式のゲームを製作するためのシステムになっています。KAG よりも作れるゲームの種類は狭くなりますが特定のゲームは効率的に製作ができます。まあ TJS を使うとそれでも色々とできるのですが。

KAGEX の方が KAG よりは用途が狭い気がする、というなんとなくの意味しかないので気にしないでください。

■「デジタル」ノベルについて

ところで本書のタイトルではアドベンチャーでもノベルでもなく「デジタルノベル」としています。デジタルノベルとはテキスト表現を中心として物語を伝えるデジタルコンテンツとして、アドベンチャーとノベルのどちらも含むものと定義しています。例によって他人には伝わらない私の個人的な定義です。ここからはノベルやアドベンチャーなどの単語を使わず、「デジタルノベル」と統一します。

デジタルノベルが好きな人はやはり物語を読むのが好きなのだと思います。デジタルノベル以外にも物語を伝えるメディアはたくさんあります。小説や漫画、アニメ、映画など枚挙にいとまがありません。これらの中で何故デジタルノベルを選ぶのでしょうか。自分が作りたいのは小説ではないのか、はたまた漫画ではないのか。製作に入る前に考えてみると時間を無駄にせずに済むかもしれません。

他のジャンルのゲームと比較すれば、デジタルノベルは少ない労力で作ることができます。同人活動など趣味の範囲でも製作され、多様な作品が生み出されています。そうは言ってもテキスト、サウンド、グラフィックなど準備しなければならない素材が多いため、複数人の大変な労力がかかります。ちゃんとしたものを作ろうとするのであれば覚悟はしておいた方がいいです。全くの初心者は他人を巻き込む前に1人で1本完成させてみるをお勧めしておきます。絵や音などの素材はインターネットで探したフリー素材でも構いません。最初からメンバーを集めて大作を作ろうとするのは挫折の元です。あなたの黒歴史に新たな1ページが加えられます。

そんな大変なデジタルノベル製作ですが、良いところも考えてみます。小説や漫画と比較すると、テキスト以外のグラフィックやサウンドなどを表現手段として使うことができる利点があります。これは漫画もですが、テキストだけではなく絵が表現の一部になります。美しいBGM や声優さんのボイスはシナリオを盛り上げ、プレイヤーの感情を揺さぶってくれます。また、映画やアニメと比較すると、テキストを使うこ

とができる点が利点になります。1枚1枚全ての場面のグラフィックを準備せずともテキストで表現すれば事足ります。テキストに小説のテクニックを導入することもできます。

プログラマの端くれとして特に言っておきたいのは、デジタルノベルにおいてはシステムも表現の一部になり得ます。コンピュータで動くゲームですので、コンピュータで実現できる表現は全て利用可能です。

「選択肢」はデジタルノベルにおいて広く採用されているデファクトスタンダードなゲームシステムです。これは決してプレイヤーにルート選択をさせるためだけの物ではありません。

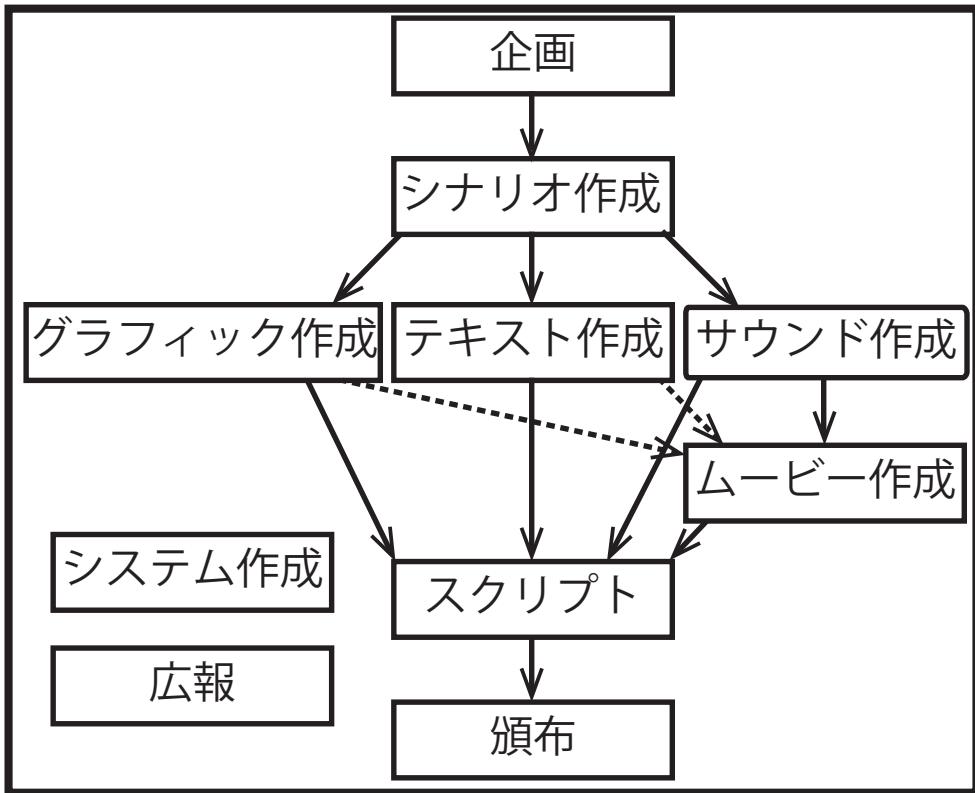
例として「07th Expansion」の『うみねこのなく頃に』を紹介します。エピソード1～8まであり、しかもそれ長いので人に勧められるようなゲームではないですが……。このゲームの最終章たるエピソード8のクライマックスにて、物語のテーマに関わる質問が選択肢の形で投げかけられます。それまで受動的に読んでいるだけだったプレイヤーが、立ち止まって能動的に考え、選択することを強いられます。ボーッっと読んでいるだけだった私には非常に印象に残っています。

もう1つ、システムで表現する例として「Key」の『リトルバスターズ！』を挙げておきます。こちらのゲームはいわゆる学園モノですが、毎日の放課後に部活としてバッティング練習というミニゲームが挟まれます。ミニゲーム中でかわいいドット絵が動く様はそれだけでキャラクターに愛着をもたせる役割を果たしています。それに留まらず、このミニゲーム自体が「みんなで練習をした」などというテキストによる表現の代替になっています。ついでにメンバーが集まっていく過程を視覚的に表したり、キャラクタの成長を表す意味もあったりと重要な要素になっています。

もしあなたがライターであれば、それは本当にテキストで表現すべきなのかちょっとと考えてもらえると嬉しいです。その点でキロバイト何円などというシナリオの仕事形態はよろしくないです。ちゃんと専任のディレクターがついてシナリオ全体でディレクションできるといいのですけれども。同人においては色々と小回りがきくと思います。おもしろいゲームが製作されることを期待しています。

■デジタルノベル製作の流れ

次のページの図はざっくりとしたデジタルノベル製作の流れです。本書で解説するのは「スクリプト」の部分です。その他はここでテキトーな説明だけしておきます。詳しくはそれぞれ専門の本を読むなりで何とかしてください。実際には人に頼むとしても、頼むなりに最低限の知識が必要になります。



前から順番に段階を踏んでいくウォーターフォールモデルになっています。しかし理想と現実というやつで、実際はこの通りになど進みません。素材が揃っていないくともスクリプトを組むなど当たり前です。できるまで待っていても仕方ないので如何ともしがたいですね。

・企画

どんなゲームを作るのか考えます。ジャンルがデジタルノベルでなければ本書とはさようならです。外注から見ても企画書は大切です。おもしろそうなゲームであれば、やはり人間ですので普段以上にやる気ができます。

・シナリオ作成

デジタルノベルの場合は、コンテンツの中核となる物語（シナリオ）がなければどうにもならないです。必要なグラフィックやサウンドなどの素材を決定するためにプロットなども書くようです。企画、シナリオ作成、テキスト作成の各段階は特に未分化であったりします。

・テキスト作成

ゲーム中で使用するテキストは多少なりともゲーム画面を思い浮かべながら書かないとスクリプトの段階で困ります。何も考えていなければテキストが枠からはみ出す、

なんてこともあったりします。この段階で、ある程度演出上の指示も入れたりします。どの背景や立ち絵を表示するかや、声優さんへの指示などになります。取りあえず小説とゲームのテキストでは全く異なるので気をつけた方がいいです。

・グラフィック作成

デジタルノベルでは背景画像の上にキャラクタの立ち絵画像を重ね、その上にテキストを表示するスタイルが定着しています。要所要所ではイベント絵(一枚絵 /CG/ スチルなどとも呼ばれます)が挟まったりします。それに加えてタイトル画面やセーブ画面で使う画像やテキスト表示部分の枠など、システム画像も必要になります。小さいものだとゲームのロゴやゲーム実行ファイルのアイコンなどもです。吉里吉里を使う場合は最終的に TLG という独自のファイル形式に変換します。ファイル形式については 3 章で解説します。



・サウンド作成

BGM、効果音、システム音(ボタンを押した時の音など)などが必要になります。場合によっては声優さんのボイス収録をしたり、主題歌のレコーディングをしたりするかもしれません。吉里吉里では wav ファイルか ogg ファイルが使用できます。一般的に使用されているサウンドファイル形式として mp3 がありますが、デジタルノベル製作ではライセンスの問題があるため使われません。

・ムービー作成

オープニングムービーなどを使う場合は必要になります。吉里吉里では MPEG1,WMV,FLASH が再生できます。

・スクリプト

製作した素材を総合して 1 つのゲームの形にしていきます。本書で主に説明する部分です。次章以降から解説します。

・頒布

ウェブページでダウンロードさせるのか、CDに焼いてイベントで配るのか、ショップで委託販売するのか、その他いろいろ。ご自由にどうぞ。

・システム作成

基本的な部分は吉里吉里 /KAGEX がやってくれています。ゲームごとに特殊な機能が必要であったり、ミニゲームを挟んだりする場合には特に必要になります。吉里吉里ではTJSないしC++などを使ってプログラミングします。

・広報

存在を知らないゲームは遊んでもらえません。重要です。ウェブページ、ブログ、Twitter、Pixiv、Facebook、ニュースサイト、イベント、雑誌、ペーパー、その他。色々あるようです。

2.開発環境の準備

この章では吉里吉里 /KAGEX の SDK など必要なファイルをダウンロードし、ゲーム製作ができる環境を整えます。

■拡張子の表示

拡張子が表示されていない環境の場合、これからの説明がわかりづらくなってしまうので前もって表示しておきます。既に表示されている環境の方は読み飛ばして構いません。取りあえずの説明はしますが、「拡張子」という言葉を初めて聞いたという方にはこの本の説明は難しいかもしれません。どうにも理解できなくなったらインターネットで検索でもして頑張ってください。

Windows では、「拡張子」というものでファイルの形式（種類）を識別しています。拡張子はファイル名の最後に「.」(ピリオド) で区切られてくっついています。例えば、「メモ .txt」というファイル名だった場合、「txt」が拡張子ということになります。「txt」はメモ帳などで開くことができる「テキストファイル」の拡張子になります。他にも「exe」「zip」「jpg」など様々な拡張子が存在します。

ゲーム開発をする上で拡張子は重要になりますが、Windows では拡張子の表示が省略されるという残念な初期設定になっています。これを省略されない設定に変更します。ここでは Windows 7 の場合の変更方法を説明します。XP や Vista などの場合は、「windows xp 拡張子 表示」などで検索すると、画像付きでわかりやすいウェブページがいくらでも出てくるので参考にしてください。

まず、左下のスタートメニューから「コントロールパネル」をクリックします。コントロールパネルが開くので、「デスクトップのカスタマイズ」→「フォルダーオプション」の順にクリックします。次に、「表示」タブを選択して「詳細設定」の中から「登録されている拡張子は表示しない」を探したて、チェックを外します。「OK」ボタンをクリックすれば拡張子が省略されないように設定が変更されます。

■吉里吉里 /KAGEX の SDK を入手

吉里吉里 /KAGEX でゲームを開発するには、吉里吉里本体の他にも KAGEX やその他のプラグインなど、色々と必要なファイルがあります。ひとつずつダウンロードしてくるのは面倒なので本書のサポートページでまとめて SDK(Software Development Kit, ソフトウェア開発キット)としてダウンロードできるようにしてあります。[「<http://biscrat.com/book/kagex/support/>」](http://biscrat.com/book/kagex/support/)にアクセスしてダウンロードして下さい。

手に入れた SDK は ZIP 形式で圧縮されています。各自で解凍してください。わか

らない方は次節の「圧縮 / 解凍」を参照してください。解凍すると「KAGEX_SDK」のようなフォルダができると思います。「readme.txt」が入っているので、クリックして読んでおいてください。

「KAGEX_SDK」フォルダ内の「吉里吉里安定版 SDK」フォルダにも「readme.txt」と「license.txt」が入っています。こちらもしっかり読んでおいてください。吉里吉里を使用する際のライセンス（利用許諾条件）になります。W.Deer 氏が製作した吉里吉里を使わせてもらうことになります。もちろん好き勝手に使う訳にはいかないので、そこに書かれている条件は必ず守らなければなりません。以下の説明は全く正式なものではなく、理解の助けになるように、かなりテキトーに説明したものです。必ず「license.txt」を自分で熟読して理解してください。

吉里吉里は「GNU GPL」または「吉里吉里独自のライセンス」のどちらかのライセンスを選んで利用することができます。ゲームを作る上では「吉里吉里独自のライセンス」の方が重要です。GPL の方が吉里吉里を利用している例は見たことがないです。「吉里吉里独自のライセンス」では吉里吉里を無償で利用することができます。製作したゲームを無料配布する場合も、お金を取って配布する場合も変わりません。ゲームとして配布する際に「吉里吉里を使っています」などと表示する義務もありません。すごい太っ腹です。C++ を使って吉里吉里本体を改造する場合は他にも条件がありますが、本書ではそんなところまで扱いません。

「吉里吉里安定版 SDK」フォルダには、ライセンスの他に「kag3」フォルダと「kirikiri2」フォルダが入っています。

「kag3」フォルダには KAG システムとその説明書などが入っています。本書では KAG ではなく KAGEX を利用するので kag3 フォルダは使用しません。説明書などを読んでみてもいいですが、KAGEX と全く異なることは意識しておいてください。下手に読むと混乱します。

「kirikiri2」フォルダには吉里吉里の本体と説明書が入っています「krkr.eXe」が吉里吉里本体になります。クリックしてみると「フォルダ / アーカイブの選択」というウィンドウが出てきます。まだ準備が済んでいないので取りあえず「キャンセル」をクリックして終了しておきます。「kr2doc」フォルダに吉里吉里本体の説明書（リファレンス）、「tjs2doc」フォルダに吉里吉里で使用する TJS の説明書が入っています。いきなり読んでも難しいですが、慣れてくるとこれらも使うようになるかもしれません。説明書の存在は頭の片隅にでも置いておくといいです。「tools」フォルダにはゲーム製作をする上で使用するツールが入っています。本書でも一部のツールを利用します。

■圧縮 / 解凍

「圧縮」とは、ファイルのサイズを小さくすることです。サイズが大きいとインターネットを通して送受信するのに時間がかかるてしまうので、インターネットで配布されているファイルの大部分は圧縮されています。圧縮の形式には ZIP 以外にも LZH、RAR、CAB、7Z など、様々なものがあります。ZIP 形式は最もメジャーな圧縮形式になります。

圧縮されたファイルは直接使えません。圧縮されたファイルを圧縮する前の状態に戻すことを「解凍」といいます。吉里吉里を使うには ZIP 形式で圧縮されたファイルを解凍する必要がありますが、解凍するには専用のアプリケーションが必要になります。ここでは有名な「+Lhaca」の使い方を紹介しておきます。

+Lhaca は [「http://park8.wakwak.com/~app/Lhaca/」](http://park8.wakwak.com/~app/Lhaca/) からダウンロードできます。ダウンロードしたファイルを実行してインストールしてください。インストールが完了すると、デスクトップに +Lhaca のアイコンができていると思います。そのアイコンに、ダウンロードした「kagex_sdk.zip」をドラッグ & ドロップすると、自動的に解凍され、デスクトップに「kagex_sdk」というファルダができます。それが解凍された吉里吉里のフォルダになります。解凍してしまえば解凍前の「kagex_sdk.zip」は不要なので削除しても構いません。

■素材ファイルの入手

スクリプトは本を読むだけで使えるようにはなりません。読みながら試せるように、本書のサンプルスクリプトで使用している画像や音などの仮素材をダウンロードできるようにしてあります。こちらも SDK と同様にサポートページからダウンロード、解凍しておいてください。

■開発用フォルダの準備

SDK には必要なファイルが揃っていますが、フォルダごとにバラバラに入っているため使いづらいので、開発用のフォルダを作成してファイルを配置します。

まずは開発用に新しいフォルダを作ります。フォルダ名は何でもいいですが、例としてデスクトップに「開発用フォルダ」という名前で作成しておきます。デスクトップでなくとも、自分で使いやすい場所ならどこでも構いません。作成したら、SDK フォルダの「ツール」フォルダに入っている「make_dev_folder.bat」をダブルクリックで実行してください。「開発用フォルダの選択」というダイアログが開くので、先ほど作成したばかりの「開発用フォルダ」を選択して「OK」ボタンを押します。これで「開発用フォルダ」に使用するファイルが全てコピーされます。

「開発用フォルダ」にコピーされた「krkr.exe」を起動してみてください。ゲーム画面が無事に表示されればOKです。

■.Net Framework 4 の準備

いくつかの開発ツールを使うには「Microsoft .Net Framework4」をインストールしておく必要があります。Microsoft のページからダウンロード、インストールしておいてください。「<http://www.microsoft.com/downloads/ja-jp/details.aspx?displaylang=ja&FamilyID=9cfb2d51-5ff4-4491-b0e5-b386f32c0992>」からダウンロードできます。

■テキストエディタの準備

スクリプトを書くには、「テキストエディタ」という種類のソフトウェアが必要です。自分で使い慣れたものがあれば、それを使ってください。注意としては、『Microsoft Word』や『一太郎』のような「ワープロソフト」では駄目です。ワープロソフトでは文字の他に画像を挿入したり、文字の大きさを変えたりという機能がありますが、スクリプトで使えるのは単純な文字のみです。そのため、文字だけを扱うのに特化したテキストエディタを使います。

「Widows 標準のメモ帳でも大丈夫です」などと省略してしまいたいところですが、本当にメモ帳を使ってしまう人がいると悲しいので説明します。メモ帳とその他の高機能なテキストエディタでは効率が段違いになります。私が使っているのは『サクラエディタ』や『Emacs』というテキストエディタになります。ここではサクラエディタの導入の仕方を紹介します。詳しい使い方などはサクラエディタのヘルプを参照してください。

まずは必要なファイルをダウンロードしてきます。サクラエディタのウェブページ「<http://sourceforge.net/projects/sakura-editor/>」から「Download」をクリックするとダウンロードが始まります。「sakura-2-0-3-1.zip」のようなZIP形式ファイルになっているので解凍してください。解凍すると「QuickStartV2.zip」と「sakura.exe」が入っています。「QuickStartV2.zip」は不要なので削除して構いません。「sakura.exe」の方は、「サクラエディタ」などのフォルダを新規作成してその中に入れておきます。

次に、『bregonig.dll』のウェブページ「<http://homepage3.nifty.com/k-takata/mysoft/bregonig.html>」から Unicode 対応版の「bron205.zip」のようなファイルをダウンロードします。番号の部分は変わっているかもしれません。こちらも ZIP 形式ですので解凍してください。解凍すると「bregonig.dll」というファイルがあるので、先ほどの「sakura.exe」と同じフォルダに移動してください。他のファイルは不要なので削除して構いません。

最後に、本書のサポートページ「<http://biscrat.com/kagex/support/>」からサクラエディタ用の吉里吉里向け設定ファイルをダウンロードします。zip形式になってるので解凍し、出てきた「krkr_sakura」フォルダを「sakura.exe」と同じ場所に移動します。

ここまでできたら、「sakura.exe」をクリックして実行してください。吉里吉里用に設定します。メニューの「設定」→「タイプ別設定一覧」→適当に使われていない設定を選択（「設定 20」など）→「インポート」→「krkr_sample」フォルダの「tjs.ini」を選択して「開く」→「はい」→「OK」→「OK」。以上でTJSスクリプト編集用の設定ができます。同様に、設定 21あたりに「インポート」から「krkr_sample」フォルダの「ks.ini」もインポートしてください。

最後に、KAGEXスクリプトファイル(拡張子がksのファイル)と、TJSスクリプトファイル(拡張子がtjsのファイル)を自動的にサクラエディタで開けるように設定します。このように拡張子ごとに使用するソフトウェアを指定することを「関連付け」と言います。文字だけでわからなければ検索するとわかりやすい解説が出てきます

「開発用フォルダ」の中の「data」フォルダの中に「startup.tjs」があります。ダブルクリックすると、「このファイルを開けません」のようなダイアログが出てくるので、「インストールされたプログラムの一覧からプログラムを選択する」にチェックを入れ、「OK」ボタンをクリックします。「ファイルを開くプログラムの選択」ダイアログが表示されたら、「参照」ボタンから「sakura.exe」を選択して「開く」ボタンを選択します。「この種類のファイルを開く時は「選択したプログラムをいつも使う」にチェックが入っていることを確認し、「OK」ボタンを押すと、サクラエディタで「startup.tjs」が編集できる状態になります。一度サクラエディタを終了して、「startup.tjs」をダブルクリックするとサクラエディタが実行されることを確認してください。同様に、「scenario」フォルダに入っている「01.ks」もサクラエディタに関連付けします。やり方は全く同じです。以上で、拡張子がtjsのファイルとksのファイルをサクラエディタで編集できるようになりました。

これはお好みですが、「設定」メニューから「タブバーを表示」をクリックすると、複数のファイルを編集する際にタブが使えるようになります。便利なので設定しておくことをお勧めします。その他にも様々な設定項目があります。サクラエディタのヘルプも見つつ自分が使いやすいように設定していくと効率が良くなっています。

3.スクリプトの基本

この章からいよいよ KAGEX スクリプトを書きはじめます。文字や画像の表示、音の再生など、基本的な機能をひと通り触ってみます。

■フォルダ構成

ゲーム開発中は「開発用フォルダ」の「data」フォルダの中身をいじっていくことになります。この data フォルダは「プロジェクトフォルダ」とも呼びます。ゲームに使用する素材やスクリプトファイルは全てこのプロジェクトフォルダに入れることになります。

そのプロジェクトフォルダの中にもたくさんのフォルダが入っています。ここで少し説明しておきます。

素材ファイルを入れるフォルダ一覧	
bgimage	背景画像を入れます。
fgimage	立ち絵画像を入れます。
evimage	イベント絵画像を入れます。
uiimage	ボタンなどのインターフェース画像を入れます。
rule	トランジションのルール画像を入れます。
thum	サムネイル用の画像を入れます。
image	その他の画像を入れます。
bgm	BGM 用の音ファイルを入れます。
sound	効果音用の音ファイルを入れます。
voice	ボイス用の音ファイルを入れます。
video	ムービーファイルを入れます。
other	その他のファイルを入れます。

スクリプトファイルを入れるフォルダ一覧	
system	KAGEX システムの TJS スクリプトが入っています。
biscrat	著書が書いた KAGEX を拡張する TJS スクリプトが入っています。
main	その他の TJS スクリプトを入れます。
init	KAGEX の設定を変更するためのファイルが入っています。
sysscн	システム関連の KAGEX スクリプトが入っています。
scenario	ゲーム本編の KAGEX スクリプトが入っています。

素材ファイルを入れるフォルダ一覧は書かれている通りです。全て1つのフォルダに入れてしまうよりは分別した方がわかりやすくなります。製作ツールもフォルダごとに区別して動作しているので、素材を入れる先のフォルダは間違えないでください。

スクリプトファイルを入れるフォルダの方で、system、biscrat、main フォルダについてはこの本では触れません。次のステップアップとしてこのフォルダの中身も使えるようになると、様々なことができるようになってきます。興味のある方は挑戦してみてください。init、sysscн、scenario フォルダの中身がこの本の主眼となります。

「data」フォルダ直下の「startup.tjs」というファイルは、吉里吉里が実行されると一番最初に実行されるスクリプトです。編集することはないので気にしないでください。その他の「default.rpf」などのファイルは、各種ツールの設定ファイルになります。直接使用することは有りません。

■テキストの表示

長い前説でしたが、ようやく KAGEX スクリプトの解説を始めます。まずはテキストを表示してみます。ゲーム本編のスクリプトは「scenario」フォルダの「01.ks」から始まります。テキストエディタで「01.ks」を開いてください。最初は以下のようなスクリプトが書かれています。

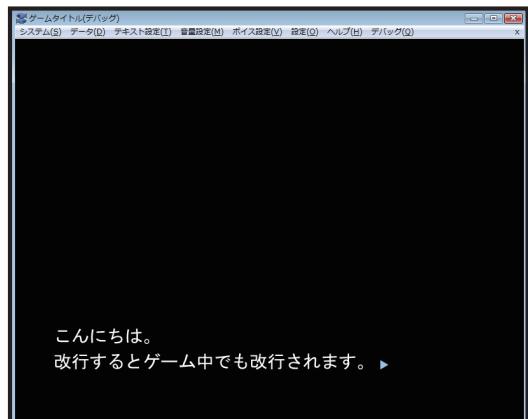
```
*start|スタート  
こんにちは。
```

最初の行の *start/スタートはラベルというものです。ラベルについては後ほど説明するのでとりあえず無視してください。次の行にはこんにちは。と書かれています。これだけでゲーム内で「こんにちは。」と表示することができます。試しに「krkr.exe」を起動すると「こんにちは。」と表示されます。

```
*start|スタート  
こんにちは。  
改行するとゲーム中でも改行されます。
```

空行を挟んだのでゲーム中でも改行されます。

テキストはただ打ち込むだけでゲーム中で表示することができます。以上のスクリプトを 01.ks に入力、保存して、krkr.exe を起動してみてください。改行もゲーム中に反映されます。また、何も書かれていない空行を挟むとクリック待ちとページ送り(表示されているテキストの消去)ができます。



*start|スタート

地の文です。

【しおり】「こんにちは。
キャラクタのセリフです。」

【しおり】

「セリフを次の行に書くこともできます。
セリフ中の改行は、カギ括弧に合わせて自動的にインデントされます。」

【】でキャラクタの名前を囲むと、発言したキャラクタの名前が表示されます。キャラクタ名は通常のテキストが表示される場所とは異なる、専用の名前欄に表示されます。「【しおり】」のあとに改行を挟み、次の行からセリフを書く場合は、「】」の後にスペースなどが混じらないように注意してください。

*start|スタート

【しおり/? ??】「名前欄には??と表示されます。」

シナリオ展開上でまだ名前が不明な場合など、キャラクタの実際の名前と異なるテキストを名前欄に表示したい場合は、「/」（半角スラッシュ）を挟みます。単純に「【??】」とすると、テキストの表示は大丈夫ですがボイスを再生するときに困ります。詳しくは5章のボイス再生のところで説明します。

*start|スタート

【しおり/? ??】「[漢字'かんじ]などに[振仮名'ふりがな]を[振'ふ]れます。」

ルビ（ふりがな）を振ることもできます。ルビを振りたい部分を[]で囲み、「（半角アポストロフィ）の後にルビとして表示する文字を書きます。

テキストの基本はこのくらいです。文字の色や大きさを変えたりなど、細かい機能は後の章で紹介します。

■吉里吉里の実行について

「開発用フォルダ」の「krkr.exe」をダブルクリックすると吉里吉里が実行できます。また、「exec_release.bat」や「exec_debug.bat」をダブルクリックしても実行できます。exec_release.batでは、吉里吉里をデバッグモードオフで実行します。開発中はデバッグモードで実行した方が便利なので、こちらはほとんど使用しません。exec_debug.batではデバッグモードオンで実行できます。開発中はこちらを使用してください。

また、スクリプトの動作を確認するたびに exec_debug.bat をクリックするのは手間がかかるので、テキストエディタのショートカットに登録しておくと便利

です。大抵のエディタには外部プログラムをショートカットに登録できる機能があると思います。

以下、サクラエディタの場合のやり方を書いておきます。詳しくはサクラエディタのヘルプを参照してください。少し面倒ですがこれから何十回と吉里吉里を起動することを思えば安いものです。他のエディタを使っている場合でもやっておいてください。

サクラエディタを起動したら、「ツール」メニューの「キーマクロの記録を開始」をクリックします。次に、同じく「ツール」メニューの「外部コマンド実行」をクリック、「...」ボタンから「exec_debug.bat」を選択し、「実行」ボタンをクリックします。吉里吉里が起動しますが、今は使わないので終了しておきます。サクラエディタに戻り、再度「ツール」メニューの「キーマクロの記録終了＆保存」をクリックすると、マクロとして保存できます。サクラエディタ本体と同じフォルダに「macro」フォルダを新規作成して、その中に「krkr.mac」という名前で保存します。

マクロが保存できたら、そのマクロをショートカットキーで実行できるようにします。「設定」メニューの「共通設定」をクリックします。「マクロ」タブを開き、「参照」ボタンから先ほど新規作成した「macro」フォルダを選択します。下の「File」の欄で「krkr.mac」を選び、名前欄に「吉里吉里起動」、Id欄を「0」にして「設定」ボタンをクリックします。次に、「キー割り当て」タブを開き、「種別」から「外部マクロ」を選択すると、「機能」欄に「吉里吉里起動」が出てくるので、それを選択します。後はショートカットとして登録するキーを選択しますが、ここでは例として(Ctrl+Q)で吉里吉里が起動できるように設定します。「Ctrl」のチェックボックスにチェックを入れ、「キー」欄から Ctrl+Q を選択します。この状態で「割付」ボタンを押せばショートカット登録は完了です。「OK」ボタンを押して共通設定ダイアログを閉じてください。登録ができたら、試しに Ctrl+Q を押すと吉里吉里が実行されることを確認してください。

■デバッグモードについて

デバッグモードで吉里吉里を実行すると、開発するのに便利な機能が使用できます。デバッグモードオンでは、ゲームウィンドウの下に「KAGEX ログ」というウィンドウが出て、クリプトに間違いがあるときに知らせてくれたりします。邪魔かもしれません、開発中は表示しておくことを推奨します。このウィンドウを閉じてしまった場合は、吉里吉里の「デバッグ」メニューの「KAGEX ログモード」をクリックすると表示できます。「デバッグ」メニューには他にもいくつか項目がありますが、いくつかの機能は後ほど紹介します。

■ショートカットキーについて

マウス操作は簡単ですが、キーボードでの操作に慣れると格段に作業のスピードが上がります。Ctrl+S(Ctrlキーを押しながらSキーを押すことです)でファイルを保存、Ctrl+Zで直前の操作取り消し、などは非常によく使うショートカットです。Ctrl+Tで半角英数字、Ctrl+Uで平仮名、Ctrl+Iで全角カタカナ、Ctrl+Oで半角カタカナ、Ctrl+Pで全角英数字に変換、などは知つてると地味に便利です。Ctrl+MはEnterの代わりになります。紹介しきれませんが他にもたくさんあります。初期設定ではショートカットキーが無くとも、よく使う機能は自分で登録しておくといいです。カーソル移動なんかもショートカット登録したりします。1日何時間もキーボードに向かっているとキーボードの右側まで指を伸ばすのも面倒なのです。カーソルキーなど時間の無駄です。そこまでやるかは別として、ゲーム製作なんて時間がいくらあっても足りないので、少しでも楽をする努力はするべきだと思います。まあ短縮された時間はやはりゲーム製作に注がれる訳ですが……。

■画像を表示する

次は画像を表示してみます。まだ仮素材として使う画像をダウンロードしていない場合は、サポートページ (<http://biscrat.com/kagex/support/>) からダウンロードしておいてください。

KAGEXで表示する画像は、背景、立ち絵、イベント絵、その他にわけられます。(テキスト枠やボタン、スライダなどシステムで使用する画像は含んでいません。それらは6章で扱います。) それぞれ使い方が異なるので、これからひとつづつ説明していきます。その前にKAGEXの画像表示の基本的な部分を説明しておきます。

■伸ばし棒の省略

コンピュータ(コンピューターに非ず)関連では単語末尾の伸ばし棒を省略することが多いです。レイヤーではなくレイヤ、スライダーではなくスライダです。大した意味は有りません。読みづらいかもしれませんがそうなってしまうので慣れてください。「プレイヤー」は例外になっています。プレイヤだと何となく違和感があったので伸ばしてます。ゲームとプレイヤーの関係というような、どちらかというと人文系の視点で見ているからかもしれません。プレーヤー? プレーヤ?などと悩んだ結果プレイヤーに統一しました。プレーヤだとmp3プレーヤみたいな意味に見えますね。本当にどうでもいい無駄話でした。

■レイヤについて

画像は「レイヤ」という透明な板に貼り付けて表示されます。フォトショップなどの画像編集ソフトのレイヤと同じようなものです。細かいことは割愛しますが、レイヤには前後関係がある、というのが重要です。後ろのレイヤに表示されている画像は、

前のレイヤに表示されている画像の後ろに隠れます。レイヤは、「背景く立ち絵とその他くイベント絵」の順番で後ろから並んでいます。背景が一番後ろ、イベント絵が一番前に表示されます。立ち絵は同時に何人も表示されたりするので立ち絵の間にも前後関係があります。詳しくは後ほど説明します。これらのレイヤのさらに上にテキスト枠とテキストが表示されます。



■画像の形式について

画像の形式には様々ありますが、吉里吉里で扱えるのは BMP、JPEG、PNG、ERI、TLG5、TLG6、WMF、EMF になります。

画像の形式には大きくわけてベクタ画像とビットマップ画像の 2 種類があります。吉里吉里で使える形式の中では WMF と EMF がベクタ画像になりますが、デジタルノベルを作る上ではまず使いません。筆者もほとんど使ったことがないため解説もできないので割愛します。残りの BMP、JPEG、PNG、TLG5、TLG6 はビットマップ画像になります。

BMP 形式はファイルサイズが大きくなりすぎるので基本的に使いません。ERI 形式というのはよく知りません。使ったこともないので無視します。

JPEG 形式と PNG 形式はそこそこ使ったりします。両者の違いは、JPEG が非可逆圧縮、PNG が可逆圧縮であるということです。非可逆圧縮では、画像がボケたりノイズが入ったりなど、汚くなってしまいます。その代わりにファイルサイズを小さく圧縮することができます。可逆圧縮では、画像は元の綺麗なままですが、ファイルサ

イズは非可逆圧縮のものと比較して大きくなってしまいます。昨今はハードディスクの容量も大きくなり、ファイルサイズが問題になることは基本的に無いので、可逆圧縮である PNG 形式を使います。何か特別に事情があって容量を小さくしなければならない時には JPEG 形式を使うかもしれません。PNG 形式にはインデックスカラーが使える、という特徴もあります。吉里吉里で「領域画像」というものを使う際にはインデックスカラーである必要があります。この本では領域画像を使わないので説明は割愛します。

TLG5 と TLG6 は吉里吉里独自の画像形式で、ほとんどの場合これを使います。どちらも可逆圧縮なので綺麗なまま画像を使えます。TLG5 は PNG 形式よりもサイズが大きくなっていますが、画像の表示が少しだけ速くなります。TLG6 は PNG 形式よりもサイズが小さくなり、TLG5 よりは遅いですが PNG よりは画像の表示も速くなります。どちらかはケースバイケースになるかもしれません、私はたいてい TLG6 を使っています。

TLG5 と TLG6 は、吉里吉里独自の形式のためフォトショップや SAI など一般的の画像編集ソフトでは使えません。まずは PNG 形式で保存しておき、吉里吉里付属のツールを使って変換することになります。

■背景の表示

まずは背景画像を表示してみます。背景のサンプルとして使うのは、サンプル素材の「bgimage」フォルダにある「bg_街_昼.png」などの画像です。フォルダの画像を全て開発用フォルダ内の「bgimage」フォルダに全てコピーしてください。

背景画像のファイル名は必ず「bg[場所名]_[時間名]」という形になります。サンプルでは場所名が「街」「向日葵」「リビング」、時間名が「昼」「夕」「夜」「共通」となっています。

背景を表示するには、スクリプトを書く前に「場所名」と「時間名」をそれぞれ定義しておく必要があります。定義には本来であれば TJS スクリプトを書く必要がありますが、簡単にするためにツールを作っています。「開発用フォルダ」の「tools」フォルダの「autosetting.bat」を実行してください。「プロジェクトフォルダの選択」というダイアログが出てきますので、「開発用フォルダ」の「data」フォルダを選択して「OK」をクリックします。「bgimage」フォルダの中の画像を見て、必要な場所と時間をそれぞれ定義してくれます。

定義ができたらスクリプトを書いて実際に表示してみます。

*start|スタート

@stage stage=街 stime=昼

場所を街、時間を昼に変更しました。

@stage stage=街 stime=夜

場所を街、時間を夜に変更しました。

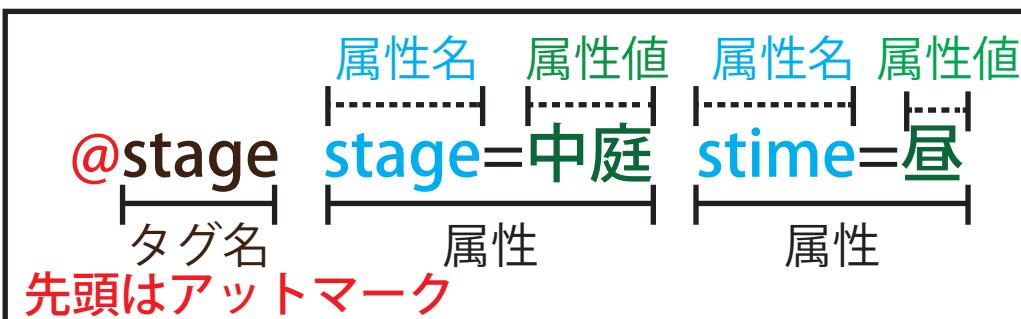
@stage stime=夕

時間を夕に変更しました。

@stage stage=向日葵

場所を向日葵に変更しました。

@ (半角アットマーク) から始まるのはコマンド行です。コマンド行はタグとも言います。画像を表示したり、音を鳴らしたりなど、何かをするには必ずタグを使います。



タグは、先頭の「タグ名」と、それに続く「属性」から成ります。`@stage stage=街 stime=昼`では、「stage」がタグ名、「stage= 中庭」と「stime= 昼」が属性です。タグ名とそれぞれの属性は半角スペースで区切られます。さらに、属性については属性名と属性値というものにわけられます。`=`(半角イコール)の前の部分が属性名、後の部分が属性値です。

背景を表示するには、「stage」という名前のタグを使います。「stage」属性に場所、「stime」属性に時間を属性値として指定します。そうすると、指定した場所と時間にゲームの状態が変更され、背景の画像もそれに合わせて変わります。`@stage stage=街 stime=昼`というタグでは、場所が街、時間が昼に変更されるので、背景には「bg 街_昼.png」が表示されます。

また、必要のない属性は省略することもあります。`@stage stime=夕`では、stage 属性を省略しているので、場所は街のままで時間だけ夕に変更されます。`@stage stage=向日葵`では、時間は夕のままで場所だけ向日葵に変更しています。

*start|スタート

@stage stage=白 stime=昼

場所を白、時間を昼に変更しました。

@stage stime=夕

時間を夕に変更しました。

背景白 _共通のままでです。

「白 _ 共通 .png」と「黒 _ 共通 .png」の時間は「共通」となっています。「共通」については、その名の通り、どの時間の場合でも共通に同じ画像が表示されます。上のスクリプトでも時間に関係なく背景は「白 _ 共通」が表示されます。

■画像の位置

背景画像は画面中央に表示されます。サンプルの「bg リビング _ 昼 .png」が画面サイズより大きいので、それで試してみます。

*start|スタート

@stage stage=リビング stime=昼

リビング、昼の背景を表示。

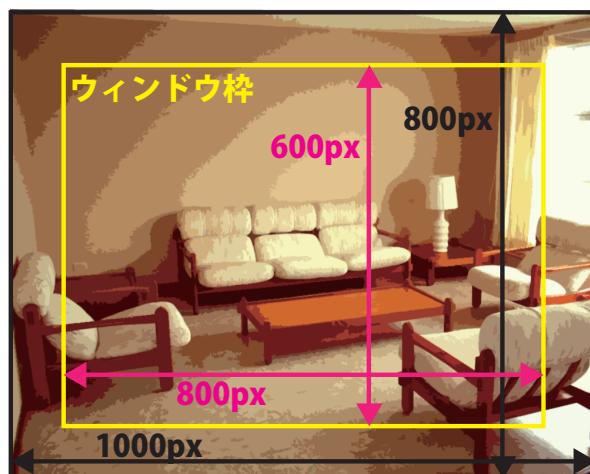
画像の中央が表示されています。

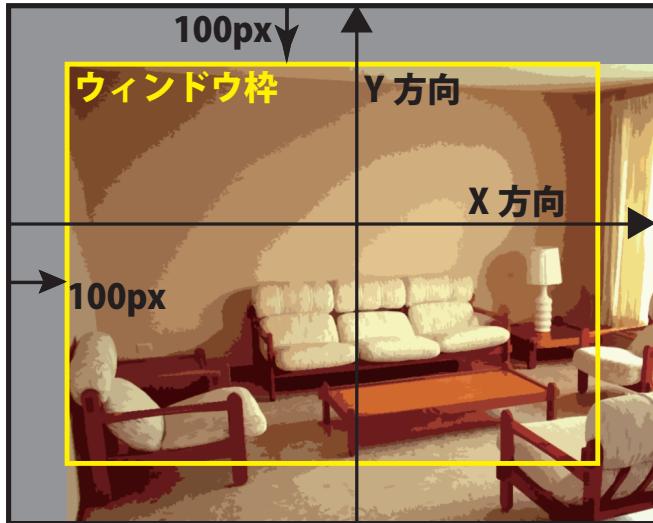
@stage xpos=100 ypos=-100

画像の位置を変更しました。

左上部分が表示されています。

デフォルトのウィンドウサイズは横 800 ピクセル×縦 600 ピクセル、リビングの画像のサイズは横 1000 ピクセル×縦 800 ピクセルです。正確にいうと背景画像は、画像の中央がウィンドウの中央になるように表示されます。なので、最初の `@stage stage=リビング stime=昼` で、下の画像の「ウィンドウ枠」の内部が表示されます。





画像の位置は、「`xpos`」属性と「`ypos`」属性で変更することができます。`xpos`が横方向で、指定した値だけ画像を右にずらせます。マイナスの値を指定すると左にずらすこともできます。`ypos`は縦方向で、指定した値だけ上にずらせます。マイナスの値を指定すると下にずらせます。`@stage xpos=100 ypos=-100`では、右に 100px、下に 100px ずらすことで画像の左上の部分を表示しています。

■ピクセル (pixel)

吉里吉里 /KAGEX での画像位置の指定は、基本的にピクセル (px) が単位になります。ピクセルはモニタで色を表示する際の最小単位です。例えば、ディスプレイの画面解像度 1920×1080 というのは横に 1920px、縦に 1080px 表示するということです。モニタに限らず、デジタル画像のサイズはピクセルが単位になっています。デジカメの何万画素とかいうのもピクセルのことです。1000px × 1000px サイズの写真が撮れるなら 1,000,000 画素になります。

ピクセルは論理的な単位なので、物理的に何センチかというのは決まっていません。同じ $100px \times 100px$ の画像でも、モニタによって表示される大きさが異なってきます。ピクセル数と実際の物理的なサイズは dpi(dots per inch) という単位で換算されます。ドット (dot) とピクセルはおなじようなものです。印刷やスキャンをするときによく使われる単位です。10 インチ × 10 インチの絵を 300dpi(1 インチに対して 300 ドット) でスキャンすると $3000px \times 3000px$ のデジタル画像になります。

吉里吉里 /KAGEX の解像度 (吉里吉里のウィンドウ内のピクセル数) は前述のとおり 800×600 になっています。ウィンドウの枠をひっぱるとウィンドウサイズが変わってウィンドウ内の実際のピクセル数も変わったりするのですが、KAGEX スクリプトを使っているうちは自動的に何とかしてくれているので意識せずに済みます。常に拡縮されていない状態で考えればいいです。TJS でカスタマイズするようになると

少し気をつける必要が出てきます。

商業ゲームだと画面サイズもどんどん大きくなっています。極端なものだと 1900 × 1200 なんてのもあったりします。同人ではまだ 800 × 600 が主流な気がします。画面サイズが大きくなると嬉しいのは、フルスクリーンでプレイしても絵が綺麗なことです。800 × 600 をフルスクリーンにすると、モニタによっては絵が汚くなってしまうかもしれません。製作側としては画像を大きく作る必要が出てくるので大変です。ウェブのフリー素材を使う場合は 800 × 600 のサイズが多かったりするのでその点もネックになるかもしれません。

もう 1 つトピックとしてアスペクト比があります。アスペクト比とは解像度の横と縦の比率のことです。800 × 600 ならアスペクト比 4:3、流行りのワイド対応なら 1280 × 720 でアスペクト比 16:9、などとなります。ワイドでは画面が横長で文字が読みづらい、絵がおかしくなるなど今までの感覚で作ると問題が出てきます。ワイド化のせいで CG が斜めになりすぎる、というネタもあったりします。(参考 : 2ch 「エロゲーは 4:3 で良い。ワイド化は誤り」スレ <http://pele.bbbspink.com/test/read.cgi/erog/1299685511/>) 参考 URL では主観視点云々という話とつながってますが、映画的に三人称視点で鑑賞している身としてもカメラが傾いているのはおかしいです。ワイド化が誤りとは言いませんがもうちょっとやりようはあるのかなーと。テキスト表示は下端から横端に移して画面を縦に使う、というのが 1 つ。スクリプトを組んでいると、立ち絵の絶対領域がテキスト枠に隠れているのがもったいない、とおもいます。何ならワイド化やめて縦長画面にしてしまってもいいのではないか。縦長画面に表示されてる女の子って、それだけでかわいさ 3 割増なのです。iPad はデバイスの画面自体が縦長でその点期待です。パソコンでも縦 1000 あれば縦長でゲーム作れるはずです。誰かやりましょう。ノートパソコンの方はごめんなさい。

関係ないですが主人公=自分だと思ってプレーしてる方って結構いるんですねえ。自分にはその発想 자체がなかったです。主人公と女の子がキャッキャウフフしているのを見ているだけ、という感覚に近い。演出付ける際も、主人公視点という意識は薄く、どちらかというと三人称視点で動かしてます。主人公の立ち絵もあればいいのに、と思うこともしばしば。

閑話休題。スクリプト解説に戻ります。

■立ち絵の表示

背景の次は立ち絵画像を表示してみます。サンプル素材の「fgimage」フォルダの「fg_しおり_ポーズ a_私服_笑顔_0.png」などの画像を使います。全て開発用フォルダの「fgimage」フォルダにコピーしてください。

立ち絵画像のファイル名は「fg[キャラ名]_[ポーズ名]_[服装名]_[表情名]_[表示レベル 〕」という形になります。サンプルでは、キャラ名は「しおり」と「かえで」、ポーズ名は「ポーズ a」と「ポーズ b」……などとなっています。ファイル名に英字を使う場合は必ず小文字にしてください。「ポーズ A」ではなく「ポーズ a」である必要があります。

立ち絵については、「キャラ名」、「ポーズ名」、「服装名」、「表情名」、「表示レベル」についてそれぞれ定義しておく必要があります。こちらも autosetting.bat で自動的に定義できます。立ち絵画像をコピーしてから、autosetting.bat を実行してください。

いま使うサンプル画像は 1 枚の画像で立ち絵が表示できます。「表情合成型立ち絵」といいます。この場合、ポーズ名、服装名、表情名の区分はあまり重要ではありません。自前の立ち絵画像を使う場合にも、特に気をつける点もなくポーズが異なっていたらポーズ名、服装が異なっていたら服装名、表情が異なっていたら表情名の部分を適当に変えておけば OK です。キャラ名、の部分は説明不要かと思います。キャラクタごとの名前になります。表示レベルは立ち絵のサイズで分類することになると思います。表示レベルは必ず 0 以上の半角数字で、数字が大きいほど立ち絵の中でも前の方に表示されます。実際のスクリプトはこれから見ていきます。

立ち絵を「土台部分の画像 + 表情部分の画像」と 2 枚の画像を組み合わせて表示する形式にすることもできます。こちらは「表情分離型立ち絵」といいます。こうすると画像の容量が小さくなったり素材管理が楽になったりします。こちらの場合の画像の作り方は後述します。一度画像を作って定義てしまえば、合成型も分離型もスクリプトでの使い方は全く同じになります。

```
*start|スタート
@stage stage=街 stime=昼
背景を表示しました。

@char name=しおり pose=ポーズ a dress=制服 face=笑顔 level=0
【しおり】「立ち絵を表示しました。」

@char name=しおり pos=消
立ち絵を消去しました。

@char name=しおり face=驚く
表情を変更してもう一度表示。
```

立ち絵を表示するには「char」タグを使います。「name」属性にキャラ名、「pose」

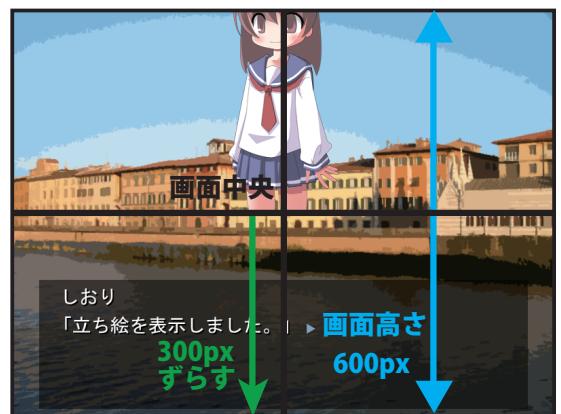
属性にポーズ名、「dress」属性に服装名、「face」属性に表情名、「level」属性に表示レベルを指定します。属性の数が多いですが、「name」属性以外は最初にまとめて指定しておくだけで、後は変更する属性のみ指定すれば十分です。「name」属性はどの立ち絵を操作するのか判定するため省略できません。

「pos」属性では立ち絵の表示状態を変更できます。「pos= 消」とすると立ち絵が非表示になります。表示する際は表情やポーズなどを変更すれば自動的に表示されます。

背景での表示、非表示の方法を説明していませんが、背景では「黒_共通.png」のような黒ベタの画像を表示させればいいので背景自体を非表示にすることはしません。

■立ち絵xオフセット定義

先ほどのスクリプトを実行してみると、左下の画像のように立ち絵が変なところに表示されます。実は立ち絵は、立ち絵画像の下端中央が画面中央になるように表示されます。何故この位置になるのか詳しい話は4章で説明します。とりあえず今回の場合は、右下の画像のように下に300pxずらすとちょうど良くなります。



ずらすには、背景と同じく xpos 属性と ypos 属性を使ってもいいのですが、全ての立ち絵の位置を一括でずらすように設定することもできます。立ち絵の最初のズレを直すにはそちらの機能を使った方が便利です。

プロジェクトフォルダの「init」フォルダの「envinit.otjs」をテキストエディタで開いてください。これは、何度か使用した autosetting.bat の動作を設定するファイルになります。開いたら、次のように書かれた部分を探してください。

```
// ■ yoffset  
// 表示レベルごとに、立ち絵の Y方向オフセット (ずらし量)を指定します。  
yoffset[0] = 0;  
yoffset[1] = 0;  
yoffset[2] = 0;
```

オフセットは、表示レベルごとに設定できます。*yoffset[表示レベル]* = オフセット ; の形式で設定していきます。初期設定では表示レベル 0、1、2 でオフセット 0(全くずらさない)となっています。サンプルで使っているのは表示レベル 0 と 1 なので、そのオフセットを下方向 300px に設定すれば OK です。yoffset では、ypos 属性とは逆に正の値を指定すると下、負の値を指定すると上にずれるようになります。

```
// ■ yoffset  
// 表示レベルごとに、立ち絵の Y方向オフセット (ずらし量)を指定します。  
yoffset[0] = 300;  
yoffset[1] = 300;  
yoffset[2] = 0;
```

このように変更すれば OK です。変更したら、tools フォルダの autosetting.bat をもう一度実行すると変更が適用されます。ゲームを起動して立ち絵の位置がずらされていることを確認してください。

■表示レベル

*start|スタート

@char name=しおり pose=ポーズ a dress=制服 face=笑顔 level=0

【しおり】「立ち絵を表示しました。」

@char name=かえで pose=ポーズ a dress=私服 face=無表情 level=1

【かえで】「もう 1人の立ち絵を表示しました。」

@char name=かえで level=0 xpos=80

【かえで】「表示レベルと表示位置を変更しました。」

@char name=しおり front

【しおり】「前に移動します。」

@char name=しおり back

【しおり】「後ろに移動します。」

立ち絵は、表示レベルが大きいほど前に表示されます。かえでの立ち絵は *level=1* で、しおりの *level=0* よりも大きいのでしおりの立ち絵の前に表示されます。その後 @char

*name=かえで level=0 xpos=80*と表示レベル 0 に変更しても、後から変更されたものが前になるため、かえでの立ち絵が前に表示されたままです。

表示レベルが同じ立ち絵の前後関係は、「front」属性と「back」属性で変更できます。*@char name=しおり front* のように属性値は省略して属性名のみで使います。front 属性で同じ表示レベルの中で一番前、back 属性で一番後ろに表示順を変更できます。

■イベント絵の表示

次にイベント絵を表示します。サンプル素材の「evimage」フォルダの「ev 発見 .png」を使います。イベント絵のファイル名は「ev[イベント名]」という形式になります。

イベント名についても autosetting.bat で定義しておく必要があります。サンプル画像をプロジェクトフォルダの「evimage」フォルダにコピーしたら、autosetting.bat を実行しておいてください。

```
*start|スタート
@stage stage=街 stime=夕
@char name=しおり pose=ポーズ a dress=制服 face=笑顔 level=0
背景と立ち絵を表示しました。

@event file=発見
イベント絵を表示しました。

@stage stage=向日葵
背景を変更しました。
```

イベント絵を表示するには「event」タグを使います。「file」属性にはイベント名を指定します。イベント絵は表示レベル 4 と表示レベル 5 の立ち絵の間に表示されるので、*@event file=発見*でイベント絵を表示すると背景と立ち絵は後ろに隠れて見えなくなります。*@stage stage=向日葵*のように stage タグで背景または時間を変更するとイベント絵は非表示になります。

イベント絵の表示位置は、背景と同じく画像の中央がウィンドウの中央となります。位置をずらすには xpos 属性と ypos 属性を使います。

■その他の画像の表示

背景、立ち絵、イベント絵以外の画像を表示したい場合があります。サンプル素材の「image」フォルダの「星 .png」を表示してみます。プロジェクトフォルダの「image」フォルダにコピーしておいてください。

その他の画像のファイル名に決まりはないので、自分でわかりやすいファイル名であれば構いません。事前に autosetting.bat などで定義しておく必要もありません。

```
*start|スタート  
@layer name=お星様 file=星 show  
星を表示しました。
```

```
@layer name=お星様 hide  
星を非表示にしました。
```

```
@layer name=お星様 show  
もう一度表示しました。
```

他の画像は「環境レイヤ」に表示します。環境レイヤは「layer」タグを使うことで表示できます。「name」属性に環境レイヤの名前、「file」属性に画像ファイル名を指定すると、指定した名前の環境レイヤに画像ファイルを読み込まれます。名前は何でもいいので適当に付けてください。名前が違うと別の環境レイヤになるので、同じ画像を複数枚表示させることもできます。環境レイヤは、属性値を省略した「show」属性で表示、「hide」属性で非表示にできます。

```
*start|スタート  
@layer name=お星様 1 file=星 show  
星を表示しました。
```

```
@layer name=お星様 2 file=星 show xpos=20 level=1  
星をもう一枚表示しました。
```

```
@layer name=お星様 1 level=2  
星 1 を前にしました。
```

環境レイヤにも、立ち絵と同じく表示レベルがあります。「char」タグではなく「layer」タグになっているだけで全く同じです。「front」と「back」属性も同じように使用できます。

背景、イベント絵と同じく環境レイヤも画像の中央がウィンドウの中央となる位置に表示されます。「xpos」と「ypos」属性で位置をずらせるのも同じです。

実は「file」属性でファイルを読み込むと自動的に表示されるので最初の `@layer name=お星様 1 file=星 show` の「show」属性は不要だったりします。わかりやすいように「show」を付けるようにしていますが面倒なので付けなくても構いません。逆に「file」

で読み込むだけで表示したくない場合は「hide」属性を付ける必要があります。

■コメント

```
*start|スタート  
;コメントのサンプルです  
;先頭が;の行は無視されます  
@stage stage=街 stime=昼  
背景を表示しました。
```

演出や演技の指示など、スクリプト中にメモ書きをしたい場合があります。「;」(半角セミコロン)を先頭につけると、その行は「コメント」としてゲームの動作に影響を与えなくなります。セミコロンがついた行は単純に無視されるので何でも書けます。

ついでに言うと、TJS スクリプトでは「//」(半角スラッシュ 2 つ)がついた行がコメントになります。envinit.otjs では//表示レベルごとに、立ち絵の Y 方向オフセット(ずらし量)を指定します。のような行がありました。コメントによるメモ書きになっています。

■省略記法

背景、立ち絵、イベント絵、その他とゲーム中で使用する画像の出し方を紹介しました。これらは非常に頻繁に使いますが、タグ名や属性が長くて大変なので省略して使えます。

```
*start|スタート  
;@stage stage=街 stime=昼と同じ  
@街 昼  
背景を表示しました。  
  
;@stage stage=向日葵と同じ  
@向日葵  
場所を変更しました。  
  
;@stage stime=夕と同じ  
@夕  
時間を変更しました。
```

背景については、「stage」というタグ名と、「stage」、「stime」の属性名を省略して、場所名または時間名をタグ名とします。xpos のような他の属性名は省略できないので @stage xpos=20 または @街 xpos=20 のようになります。

```

*start|スタート
;@char name=しおり pose=ポーズ a dress=制服 face=笑顔 level=0
@しおり ポーズ a 制服 笑顔 奥
【しおり】「立ち絵を表示しました。」

;@char name=しおり pose=ポーズ a dress=私服 face=無表情 level=1
@かえで ポーズ a 私服 無表情 前
【かえで】「もう一人立ち絵を表示しました。」

;@char name=かえで face=笑顔
@かえで 笑顔
【かえで】「表情を変更しました。」

;@char name=かえで pos=消
@かえで 消
立ち絵を消去しました。

```

立ち絵については、「char」というタグ名と、「name」、「pose」、「dress」、「face」、「level」、「pos」の属性名を全て省略できます。「name」属性に指定していたキャラクタ名をタグ名として使えます。

表示レベルを指定する「level」属性を省略するには、表示レベルの名前を定義しておく必要があります。init フォルダの envinit.otjs を開いて、以下のようなところを探してください。

```

// ■ levels
// 表示レベルごとに、立ち絵の表示レベルに名前をつけます。
levels[0] = "奥";
levels[1] = "前";
levels[2] = "手前";

```

`levels[表示レベル] = "表示レベルの名前";` という形で定義します。表示レベルの名前は ""(半角ダブルクオート)で囲みます。デフォルトでは「奥」が `level=0`、「前」が `level=1`、「手前」が `level=2` の省略として使えます。必要なら書き換えて autosetting.bat を実行すると、定義した名前がスクリプト中で使えるようになります。本書のサンプルは変更せずこのままで使います。

立ち絵については、「xpos」属性にも名前を定義して省略することができます。こちらも「envinit.otjs」を見てください。

```
// ■ xpos  
// X方向の位置に名前をつけます。  
xpos.左外 = -400;  
xpos.左奥 = -300;  
xpos.左 = -200;  
xpos.左中 = -100;  
xpos.中 = 0;  
xpos.右中 = 100;  
xpos.右 = 200;  
xpos.右奥 = 300;  
xpos.右外 = 400;
```

xpos.名前=xposの値; の形で定義します。デフォルトでは「左外」が *xpos=-400*、「左奥」が *xpos=-300*、「左」が *xpos=-200*……の省略となっています。こちらも必要なら書き換えて、「autosetting.bat」を実行しておいてください。本書のサンプルではこのままで使います。

```
*start|スタート  
@しおり ポーズ a 制服 笑顔 奥 左  
【しおり】「立ち絵を表示しました。」  
  
@かえで ポーズ a 私服 無表情 奥 右  
【かえで】「もう一人立ち絵を表示しました。」
```

```
@しおり 消  
@かえで 消  
立ち絵を消去しました。
```

実際の使い方は上のようになります。服装や表情と同じく並べて指定するだけです。

```
*start|スタート  
; @event file=発見 同じ  
@発見  
イベント絵を表示しました。
```

イベント絵については、「event」というタグ名と「file」属性を省略できます。イベント名がタグ名になります。位置を変更する際は *@発見 xpos=30* のようになります。イベント絵であることが分かりやすいように *@ev発見* とタグ名に「ev」を付けることもできます。使いやすい方を使ってください。

```
*start|スタート  
@layer name=お星様 file=星 show  
星を表示しました。
```

```
; @layer name=お星様 hide と同じ  
@お星様 hide  
星を非表示にしました。
```

```
; @layer name=お星様 show と同じ  
@お星様 show  
もう一度表示しました。
```

環境レイヤについては、タグ名の「layer」と、「name」属性を省略できます。「name」に指定していた環境レイヤ名がタグ名になります。

ただし、初めて使う環境レイヤ名の場合は省略することができません。初めて使う際に指定した名前の環境レイヤが作られるので、既に作られている場合にのみ省略することができます。

```
*start|スタート  
; ここで「お星様」という名前の環境レイヤが作られます  
; @お星様 file=星 show と省略はできません  
@layer name=お星様 file=星 show  
星を表示しました。
```

```
; @layer name=お星様 xpos=100 と同じ  
@お星様 xpos=100  
星の位置を変更しました。
```

```
; @layer name=お星様 delete と同じ  
@お星様 delete  
環境レイヤ自体を削除しました。
```

```
; 次に使うときは再び「@layer」で環境レイヤを作る必要があります  
@layer name=お星様 file=星 show  
星を表示しました。
```

「hide」属性で非表示にするのではなく、`@お星様 delete`と「delete」属性を使うことで作成された環境レイヤを削除することもできます。属性値は省略します。使わなくなった環境レイヤが残るのが気になる場合は削除してもいいかもしれません。削除し

た場合は、次に使う際に再び省略せずに「@layer」を使って環境レイヤを作成させる必要があります。

■BGM の再生

画像はひと通りやったので、BGM を再生してみます。

サンプルの「bgm」フォルダの「bgm 湖畔の村 .ogg」と「bgm 冬の息吹 .ogg」をプロジェクトフォルダの「bgm」フォルダにコピーしてください。bgm は、「bgm[bgm 名]」という形のファイル名になります。BGM については autosetting.bat などを実行する必要はありません。

```
*start|スタート  
@bgm play=bgm湖畔の村  
BGMを再生しています。  
  
@bgm play=bgm冬の息吹 loop=false  
BGMを切り替えました。ループ再生はしません。
```

```
@bgm stop  
BGMを停止しました。
```

BGM を再生するには「bgm」タグを使用します。「play」属性に BGM ファイル名を指定すると、そのファイルが BGM として再生されます。BGM が最後まで再生し終わると自動的に最初からループ再生されます。「loop」属性に「false」を指定するとループ再生しなくなります。属性値を省略した「stop」属性で BGM の再生を停止できます。

```
*start|スタート  
;@bgm play=湖畔の村と同じ  
@bgm湖畔の村  
BGMを再生しています。  
  
;@bgm play=冬の息吹 loop=falseと同じ  
@bgm冬の息吹 loop=false  
BGMを切り替えました。ループ再生はしません。  
  
@bgm stop  
BGMを停止しました。
```

BGM については再生する際にタグ名の bgm と play 属性を省略できます。`@bgm湖畔の村`のように BGM ファイル名をタグ名とするとその BGM が再生されます。

■音楽ファイルの形式について

吉里吉里で主に再生できる音楽ファイルの形式は、PCM WAV(拡張子は wav)と OggVorbis(拡張子は ogg)になります。

wav は音楽 CD でも使われている形式で、基本的に無圧縮のため音質は良いですがファイルサイズが大きくなってしまいます。一般的な設定の wav で 1 分 10M ほどになります。ファイルサイズが大きくなつてもいい場合は使うこともあります。

ogg は非可逆圧縮のため音質は悪くなりますがファイルサイズを小さくすることができます。ゲーム中では大抵の場合こちらを使います。一般に使用されている mp3 とは異なりライセンス料がかかりません。wav や mp3 など、他の形式から ogg に変換するツールは検索すると色々でてくるので好きなものを使ってください。

ogg や mp3 は非可逆圧縮のため、ファイルを編集するごとに音質がどんどん劣化していきます。エフェクトをかけたり音量調整をしたりなどは必ず元の wav ファイルでやるようにしてください。編集が終わってから、最後に ogg に変換してゲーム中で使用します。

実は 5.1ch サラウンドにも対応しているようです。誰かやってみたら楽しいかも。

wav と ogg の他にも MIDI、CD-DA、独自形式の TCWF などが使えるらしいです。が、使わないので解説しません。

■効果音の再生

次に効果音 (SE = Sound Effect) を再生してみます。サンプルの「sound」フォルダの「se ボタン.ogg」と「se 雨.ogg」をプロジェクトフォルダの「sound」フォルダにコピーしてください。効果音は「se[効果音名]」という形のファイル名になります。効果音についても autosetting.bat などを実行する必要はありません。

```
*start|スタート  
@se play=se雨 loop=true  
SEをループ再生しています。
```

```
@se play=seボタン  
もう 1 つの SE をワンショット再生しました。
```

```
@se name=se雨 stop  
SEを停止しました。
```

効果音を再生するには「se」タグを使用します。「play」属性に効果音ファイル名を指定すると、そのファイルが効果音として再生されます。効果音はBGMと異なりそのままではループ再生されません。ループせずに1回だけ鳴らすことをワンショット再生と言ったりもします。「loop」属性に「true」を指定すると効果音でもループ再生させることができます。

BGMと効果音での大きな違いは、BGMが同時に1曲しか再生できないのに対し、効果音は同時にいくつも再生することができます。上のスクリプトでは「se 雨」をループ再生している間に「se ボタン」も再生しています。同時に再生できる効果音の数はデフォルトでは3つになっています。設定で増やせますが、4つ以上同時再生したいケースは稀です。

再生を停止する場合は「stop」属性を使用しますが、同時にいくつも再生できる効果音の場合は「name」属性でどの効果音を停止するかを指定する必要があります。`@se name=se雨 stop` のようになります。

```
*start|スタート  
;@se play=se雨 loop=true と同じ  
@se雨 loop=true  
SEをループ再生しています。
```

```
;@se play=seボタン と同じ  
@seボタン  
SEをもう一つ再生しました。
```

```
;@se name=se雨 stop と同じ  
@se雨 stop  
SEを停止しました。
```

seについてはタグ名のseとplay属性を省略できます。効果音ファイル名をタグ名とするとそのファイルが効果音として再生されます。停止する場合にも、name属性を省略し、ファイル名をタグ名としてstop属性をつけると停止できます。

■trueとfalse

BGMとSEに共通の属性であるloop属性には「true」または「false」を指定します。日本語ではtrue = 真、false = 偽などと訳されます。trueまたはfalseを指定する属性は他にもたくさんあります。trueを指定するとその属性の機能がオンに、falseを指定するとオフになる、と思っておくといいです。「loop」属性ではtrueを指定するとループ再生がオンに、falseを指定するとループ再生がオフになります。先ほどのス

クリプトでは BGM では *loop=false*、SE では *loop=true* しか使っていませんが、BGM で *loop=true*、SE で *loop=false* と指定することもできます。ただし、前述のとおり BGM ではデフォルトでループ再生がオン、SE ではオフになっているのでわざわざ指定する意味はありません。

■属性値 true の省略

属性は = で区切られた属性名と属性値から成りますが、属性値が true の場合は属性値を省略することができます。`@se 雨 loop` は `@se 雨 loop=true` と同じです。立ち絵の表示順を変更する際に front 属性や back 属性の属性値を省略したのを覚えているでしょうか。実はこれらも *front=true* や *back=true* を省略したものです。前に持ってくる機能をオンにする、後ろに持ってくる機能をオンにする、といった感じの意味になります。true を指定できるということは *front=false* または *back=false* と指定することもできるのですが、この場合は何も起きないので全くの無駄になります。結局 *front* または *back* と属性値を省略して使うことになります。*layer* タグでの show 属性と hide 属性についても全く同じことです。

■スクリプトの実行順

KAGEX スクリプトはほとんどタグからできています。ここまで stage、char、event、……などのタグを紹介してきました。テキストはそのまま書くだけで表示されていますが、実はこれもタグだったりします。テキストの表示には「ch」というタグが使われています。テキストは大量に使うのに、いちいちタグを書いていては面倒なので省略してそのまま書けるようになっています。改行やページ送りなども、それぞれタグが使われています。

KAGEX スクリプトは上から順番に実行されます。当たり前ですが重要です。

```
*start|スタート
@stage stage=街 stime=昼
背景を表示しました。

@char name=しおり pose=ポーズ a dress=制服 face=笑顔 level=0
立ち絵を表示しました。
```

まず背景を表示する `@stage stage=街 stime=昼` が実行されます。表示し終わったら背景を表示しました。という文字を表示する「ch」タグが実行されます。次に空行が挟まっているので、「プレイヤーがクリックするのを待つタグ」が自動的に実行され、スクリプトの実行が一時停止します。

ここでスクリプトの実行が一時停止されている、というのも知っておくといいです。

何かしらのタグで停止または一時停止されない限り、KAGEX スクリプトは上からどんどん実行されていきます。

プレイヤーがクリックするとスクリプトの実行が再開され、「ページ送りをするタグ」が自動的に実行されてテキストが消去されます。その後 `@char name=しおり pose=ポーズ` `dress=制服` `face=笑顔` `level=0` で立ち絵が表示され、最後に「ch」タグで立ち絵を表示しました。と表示されます。最後まで実行が終わったのでスクリプトの実行は停止します。

選択肢でシナリオが分岐したりなど、上から順番に実行しているだけでは済まない場合があります。選択肢 A と B が選ばれた場合でそれぞれスクリプトを実行する位置を変えなければなりません。

```
*start|スタート
```

スタートです。

```
@next target=*ラベル 1
```

```
*ラベル 2
```

ラベル 2です。

```
@next target=*start
```

```
*ラベル 1
```

ラベル 1です。

```
@next target=*ラベル 2
```

まずは選択肢などを使わずに、単純にスクリプトの実行位置を変えてみます。それには「next」タグを使用します。「target」属性にラベル名を指定すると、次はそのラベルからスクリプトが実行されます。ラベルとは `*start` や `*ラベル 1` のように `*(半角アスタリスク)` から始まる行です。ラベルは `next` タグに限らず、スクリプトの実行位置を変更するための目印の役割を果たします。もし「01.ks」の 100 行目、などと行数で指定しようとすると、タグを付け足して 1 行ずれるだけで実行したい位置もずれていってしまって大変です。ラベルを使うことで行数がずれても大丈夫なようになっています。

今まで説明してこなかった `*start`/スタートもラベルです。後ろに `|`(半角縦線)がついていますが、これは無視してください。また後回しになってしまいますがすぐに説明します。ラベル名は `|` の前の「start」になります。吉里吉里が起動されると、KAGEX

の準備やら何やらが終わってから、「01.ks」の `*start` ラベルから KAGEX スクリプトの実行が始まる、という仕組みになっています。

上のスクリプトでは、スタートです。が表示された後、`@next target=*ラベル 1` によって、`*ラベル 1` へとスクリプトの実行位置が変更されます。ラベル 1 へ「ジャンプ」する、とも言います。よって、ラベル 1 です。が表示され、同じく `*ラベル 2` へとジャンプし、ラベル 2 です。が表示されます。最後に `@next target=*start` で再び `*start` から実行されるので「スタートです。」「ラベル 1 です。」「ラベル 2 です。」……がクリックするたびに延々と表示され続けることになります。

ラベル名には「start」や「ラベル 1」のように好きな名前をつければ、スペースは使えません。

```
*start|スタート
```

スタートです。

```
@next target=*ラベル 1 storage=02.ks
```

「storage」属性を使うと、別のファイルのスクリプトを実行することができます。上のスクリプトは今までどおり「01.ks」に書いてください。

プロジェクトフォルダの「scenario」フォルダに新しいテキストファイルを作成して、拡張子も含めて「02.ks」に名前を変更します。変更したら以下のスクリプトを「02.ks」に書いてください。

```
*ラベル 1
```

02.ksのスクリプトが実行されています。

```
@next target=*start storage=01.ks
```

ゲームを起動すると、いつもどおり「01.ks」の `*start` から実行が始まります。スタートです。が表示された後、`@next target=*ラベル storage=02.ks` によって、「02.ks」の `*ラベル 1` へとジャンプします。そして `@next target=*start storage=01.ks` によって「01.ks」の `*start` へと戻ってきます。

ファイル名は 01.ks や 02.ks など「番号 +.ks」としていますが、番号にしなければならないという決まりはありません。「しおり編.ks」や「エピローグ.ks」などわかりやすい名前をつけることができます。`@next storage=しおり編.ks target=*? ? ?` のように `storage` 属性に指定すればそのファイルのスクリプトを実行できます。

■セーブデータ名の指定

*start|スタート

セーブデータ名は『スタート』です。

メニューからセーブしてみてください。

*ラベル0|0 0 0 1

セーブデータ名は『0 0 0 1』です。

*ラベル1

ここでもセーブデータ名は『0 0 0 1』です。

*|0 0 0 2

このセーブデータ名は『0 0 0 2』です。

ラベルの「|」(半角縦棒)の後ろはセーブデータの名前になります。ゲームを起動して、メニューの「データ」の「セーブ」からセーブしてみてください。どの時点でセーブするかによって「ロード」から見れるデータ名が変わっていることがわかると思います。

*ラベル0/0 0 0 1を通過するとセーブしたときのデータ名が「0 0 0 1」に変わります。次の *ラベル1 ではセーブデータ名がついていません。この場合は変更されず「0 0 0 1」のままで。*/0 0 0 2では、ラベル名が省略されています。この場合は target 属性に指定するラベル名がわからないため、next タグの実行位置の目印としては使えません。セーブデータ名のみを変えたい場合にはこのような省略ができます。

*start|スタート

セーブデータ名は『スタート』です。

ラベルがついていない次のテキストです。

その次のテキストです。

next タグの際に説明したように、スクリプトの実行位置を変更するにはラベルを目印とする必要があります。これはセーブしたデータをロードするときにも同じです。なので、ラベルがついていない次のテキストです。でセーブしたデータをロードする場合、直接そこからスクリプトを実行させることができません。KAGEX では、*start/スタートへジャンプしておいて、そこから実際にセーブしたラベルがついていない次のテキストです。までスクリプトの実行を早送りする、という方法でロードします。

2、3 行程度の早送りであれば一瞬で終わりますが、1000 行、2000 行と早送りを

するには時間がかかってしまいます。このため、前のラベルがあまりに遠いところでセーブしたデータをロードすると、早送りに時間がかかって少し止まったような動作になってしまいます。

*start|スタート

セーブデータ名は『スタート』です。

*|

ラベルがついていない次のテキストです。

*|

その次のテキストです。

これを防ぐため、他に必要が無くてもところどころにラベルを挟むようにします。私の場合は 300 ~ 500 行に 1 つはラベルを置くようにしてあります。このラベルは next タグで飛んでくる必要も、セーブデータ名を変える必要もないでの *(半角アスタリスクと半角縦棒のみ)まで省略してしまえます。ラベル名を省略すると next タグでジャンプすることはできませんが、ロード時のジャンプでは使えるのでこれで大丈夫です。

■セーブデータの消去

セーブしたデータは、「krkr.exe」と同じフォルダの「savedata」フォルダに保存されています。このフォルダの中身を削除するとゲームのセーブデータを削除できます。「krkr.exe」と同じフォルダの「clean.bat」を実行すると savedata フォルダの中身を削除してくれます。

開発中に何かおかしい、と思ったらとりあえずセーブデータを削除してみると直るかもしれません。KAGEX スクリプトのラベル名が変わったりすると、そもそもそのデータは使えなくなります。そうでなくとも、ゲームが終了する際には色々なデータが自動的に保存され、次にゲームを起動したときにそれらが読み込まれます。開発中で設定を変更したりしているとデータとのズレが生じて、設定の変更が反映されなかったり、動作がおかしくなったりする可能性があります。

ウィンドウ枠を引っ張ってウィンドウサイズを変更してからゲームを再起動してみると、データが保存されているというのがわかるとおもいます。再起動してもウィンドウサイズが変更された状態になっています。そこで「clean.bat」を実行してデータを削除し、もう 1 度ゲームを実行してみるとウィンドウサイズが戻っています。

大抵の場合はデータを削除しなくても大丈夫なのですが、何故か上手くいかないと

思つたらデータを削除してみてください。それで直つたらおめでとうございます。直らなかつたらどこか別に問題があります。

■選択肢

*start|スタート

選択肢を使ってみます。

好きな方を選択してください。

```
@seladd text=最初に戻る target=*start
```

```
@seladd text=選択肢 1 target=*選択肢 1
```

```
@seladd text=選択肢 2 target=*選択肢 2
```

```
@select
```

*選択肢 1

選択肢 1が選ばれました。

```
@next target=*start
```

*選択肢 2

選択肢 2が選ばれました

```
@next target=*start
```

選択肢は、「seladd」タグで選択肢を登録→「select」タグで登録した選択肢を表示、という流れになります。

「seladd」タグでは「text」属性に選択肢として表示する文字、「target」属性にその選択肢が選択されたときにジャンプする先のラベル名を指定します。next タグと同じく「storage」属性を使うことで他のファイルのラベルへジャンプさせることもできます。

「select」タグでは、それまでに seladd タグで登録された選択肢を表示し、プレイヤーが選択肢を選択するまでスクリプトの実行を停止します。選択されたら、その選択肢で指定されているラベルからスクリプトの実行を再開します。

■空白を含む属性値

選択肢に「スペースのある 選択肢」と表示したいとします。単純に@seladd text=スペースのある 選択肢とすると、スペースの部分で属性の区切りとなってしまい、正しく動作しません。「text=スペースのある」と「選択肢」という 2 つの属性と解釈されてしまいます。

このように属性値にスペースを含む値を指定したい場合、""(半角ダブルクオート)で囲むと正しく動作します。具体的には @seladd text="スペースのある 選択肢" のようになります。seladd タグに限らず、どのタグでも同じように使えます。さらに言えば、スペースを含まない属性値を "" で囲んでもかまいません。@next storage="01.ks" target="*start" でもしっかりと動作します。囲んだほうが見やすいのであれば囲んでください。本書では "" をタイプするのが面倒なので必要のない限り囲みません。

4.画像操作スクリプト

3章では画像の表示、非表示のやり方くらいしか説明していませんが、この章ではもう少し見た目を豪華にするスクリプトを紹介していきます。

■トランジションの基本

3章では画像は瞬時に出たり消えたりしていました。大抵のデジタルノベルではじわじわとフェードインしたりフェードアウトしたりしています。それをやってみます。吉里吉里では「トランジション」と呼ばれる機能です。

```
*start|スタート  
@街 昼 trans=crossfade time=2000  
背景をフェードインしました。  
  
@しおり ポーズ a 制服 笑顔 前 trans=crossfade time=1000  
【しおり】「立ち絵をフェードインしました。」  
  
@しおり 消 trans=crossfade time=500  
立ち絵をフェードアウトしました。
```

トランジションをするには「trans」属性を使います。trans 属性に「crossfade」を指定すると最も基本的なフェードイン、アウトである「クロスフェードトランジション」ができます。単純に、出てきたり消えたりします。

「time」属性でトランジションにかける時間をミリ秒(ms) 単位で指定します。「1ミリ秒 = 1秒の 1000 分の 1」です。*time=1000* とすると 1000ms = 1秒でトランジションします。*time=500* では 500ms = 0.5 秒になります。

trans 属性を使うと、トランジションが終了するまでスクリプトの実行が一時停止されます。背景が完全に表示されてから背景をフェードインしました。というテキストが表示されます。このような動作を「同期動作」といいます。

同期動作に対して「非同期動作」という概念があります。非同期動作はテキスト表示と同時進行でトランジションなどが実行される動作です。非同期動作でない動作が同期動作になります。とりあえず使ってみます。

```
*start|スタート
@街 昼 trans=crossfade time=2000 nosync
背景を非同期フェードインしています。

@しおり ポーズ a 制服 笑顔 前 trans=crossfade time=1000 nosync
【しおり】「立ち絵を非同期フェードインしています。」

@しおり 消 trans=crossfade time=1000 sync
立ち絵を同期フェードアウトしました。

; 何度か試せるように最初にジャンプしています
@next target=*start
```

属性値を省略した「nosync」属性をつけると非同期動作、「sync」属性をつけると同期動作になります。背景や立ち絵のトランジションと同時に、テキストの表示も行われているのがわかるでしょうか。「テキスト設定」メニューの「テキスト速度」を遅くして試してみるとわかりやすいかもしれません。

最後の @しおり 消 trans=crossfade time=1000 sync は sync 属性がついているので先ほどと同じ同期動作です。トランジションについてはデフォルトで同期動作となっているので「sync」属性はつけてもつけなくとも変わりません。

非同期動作の実際の仕組みとしては、@街 昼 trans=crossfade time=2000 nosync でトランジションが開始されたら、トランジションの終了を待たずにそのままスクリプトの実行を継続します。なので背景を非同期フェードインしています。がそのまま実行され、空行でのクリック待ちで一時停止します。

同期動作では、トランジションを開始したらそれが終了するまでスクリプトの実行が一時停止され、終わったら実行再開、となっています。なのでトランジションとテキストの表示は同時には行われません。

同期動作においてトランジションの終了を待っている間にクリックされると、トランジションを強制的にキャンセルしてスクリプトの実行を再開します。非同期動作の場合は、ページ送りの時点でまだトランジション中であれば強制的にキャンセルされます。トランジションが「キャンセル」されると瞬時に終了後の状態になります。フェードインなら瞬時に表示され、フェードアウトなら瞬時に消えることとなります。

*start|スタート

@街 昼 trans=crossfade time=15000 nosync

背景を 15秒かけて非同期フェードインしています。

非同期動作の途中でページ送りが発生すると、非同期動作はキャンセルされます。

@しおり ポーズ a 制服 笑顔 前 trans=crossfade time=15000 nosync nowait

【しおり】「15秒かけて非同期フェードインしています。」

nowait属性をつけるとページ送りでもキャンセルされません

@しおり stoptrans

強制的にトランジションをキャンセルしました。

「nowait」属性をつけた非同期トランジションはページ送りでキャンセルされなくなります。とはいっても、いつまでも実行されていて困るので「stoptrans」属性をつけると、その時点で nowait 属性がついた非同期トランジションもキャンセルできます。キャンセルされるのは「stoptrans」属性をつけた画像のみなので注意してください。@しおり stoptrans では、仮に背景が非同期トランジションしていたとしてもそちらはキャンセルされません。属性としてではなく、@stoptrans とタグとして使えば全てのトランジションをキャンセルすることができます。

*start|スタート

@begintrans

@街 昼

@しおり ポーズ a 制服 笑顔 前 右

@endtrans trans=crossfade time=2000

背景と立ち絵を同時に同期トランジションしました。

@かえで ポーズ a 私服 無表情 前 左 trans=crossfade time=1000 nosync

@しおり ポーズ b trans=crossfade time=1000 nosync

立ち絵を 2枚同時に非同期トランジションしています。

@begintrans

@黒

@かえで 消

@しおり 消

@endtrans trans=crossfade time=1000 sync

背景と立ち絵を同期トランジションで消去しました。

背景と立ち絵を同時にトランジションする際は「begintrans」タグと「endtrans」タグを使います。トランジションで変更したい内容を @begintrans と @endtrans の間に書きます。間に書いた画像関連の変更は、書いた時点では反映されず、endtrans タグに指定したトランジションで一気に更新されることになります。begintrans タグには属性がありません。@endtrans trans=crossfade time=2000 のようにトランジションを指定することになります。

背景を含まず、立ち絵を 2 枚以上同時に「非同期」トランジションしたい場合は、begintrans と endtrans を使わずにそのまま並べてしまっても OK です。

```
*start|スタート
@begintrans
@街 昼
@しおり ポーズ a 制服 笑顔 前 右
@endtrans trans=crossfade time=5000 sync
背景と立ち絵を同期トランジションで表示しました。
```

```
@begintrans
@しおり ポーズ b
@かえで ポーズ a 私服 無表情 前 左
@endtrans trans=crossfade time=2000
立ち絵を 2枚同時に同期トランジションしました。
```

```
@しおり 私服 怒り trans=crossfade time=1000 nosync
@かえで 笑顔 trans=crossfade time=1000 nosync
@wt
立ち絵を 2枚同時に同期トランジションしました。
```

背景を含まずに複数の立ち絵を同時に「同期」トランジションする場合は、begintrans タグと endtrans タグを使ってもいいですし、「wt」タグを使う方法もあります。@wt では同期 / 非同期に関わらず全てのトランジションが終了するまでスクリプトの実行を一時停止します。上のスクリプトの最後のように、非同期トランジションを開始してすぐに終了を待てば同期トランジションと同じ動作をさせることができます。

背景と立ち絵を同時にトランジションする場合は、バラバラにトランジションすると表示が変になってしまうので begintrans と endtrnas をつかってまとめてトランジションするようにします。

wt タグでは「canskip」属性が使えます。true を指定するとクリックでトランジションをキャンセルできます。デフォルトの動作は *canskip=true* です。false を指定するとクリックしてもトランジションはキャンセルできず、トランジションが終了してからスクリプトの実行が再開されます。sync 属性による同期動作では「canskip」属性は使えないのに必ずクリックでキャンセルできます。

■他のトランジション

trans 属性には crossfade 以外にも色々と指定できます。1つずつ紹介していきます。

サンプル素材フォルダの「rule」フォルダの「rl うずまき .png」と「rl 右から左 .png」を、プロジェクトフォルダの「rule」フォルダにコピーしてください。

```
*start|スタート  
@街 昼 trans=universal time=3000 rule=rlうずまき vague=0  
背景を同期トランジションしました。
```

```
@街 夜 trans=universal time=3000 rule=rlうずまき vague=128 nosync  
背景を非同期トランジションしています。
```

```
@begintrans  
@向日葵 昼  
@しおり ポーズ a 私服 怒り 前 左  
@layer name=お星様 file=星 show  
@endtrans trans=universal time=3000 rule=rl右から左 vague=64  
【しおり】「同期トランジションしました。」
```

trans 属性に「universal」を指定すると「ユニバーサルトランジション」になります。time 属性にはトランジションの時間を指定します。「rule」属性にはモノクロのグレースケール画像のファイル名を指定します。ユニバーサルトランジションでは rule 属性に指定された画像（「ルール画像」と言います）の黒い部分から順に入れ替わっていき、白いところが最後に入れ替わります。「vague」属性はあいまい領域値というので、大きな値を指定するほど入れ替わりの境界線がぼやけてなめらかなトランジションになります。

言葉で言ってもよくわからないので数値を変えて試してみるのがいいと思います。*vague* 属性は省略でき、その場合は *vague=64* を指定したとみなされます。

ルール画像は「rl」から始まるファイル名で「rule」フォルダに入っている必要があります。

```
*start|スタート  
@向日葵 昼 trans=scroll time=3000 from="top" stay="nostay"  
背景を同期トランジションしました。
```

```
@向日葵 夕 trans=scroll time=3000 from="left" stay="stayback" nosync  
背景を非同期トランジションしています。
```

```
@ev発見 trans=scroll time=3000 from="bottom" stay="stayfore"  
【しおり】「同期トランジションしました。」
```

trans 属性に「scroll」を指定すると、「スクロールトランジション」になります。画像がスクロール（スライド）して切り替わるトランジションです。time 属性にはトランジションの時間を指定します。「from」属性は、どちらから変更後の画像がスライドしてくるかを指定します。「left」「top」「right」「bottom」のいずれかを指定し、それぞれ左、上、右、下からスライドしてきます。「stay」属性には「nostay」「stayback」「stayfore」のいずれかを指定します。「nostay」では、切り替わる前と後の画像のどちらもスライドします。ちょうど切り替わる前の画像が後の画像に押し出されるようにスライドされます。「stayback」では切り替わる前の画像がスライドして出ていき、その下から切り替わった後の画像が出てきます。「stayfore」では、切り替わった後の画像がスライドして出てきて、切り替わる前の画像にかぶさってきます。こちらもよくわからないので実際に試してみることをお勧めします。

```
*start|スタート  
@begintrans  
@向日葵 昼  
@しおり ポーズ a 私服 驚く 奥 左  
@endtrans fade=2000  
【しおり】「背景と立ち絵を同期トランジションしました。」
```

```
@しおり 消 fade=1000 nosync  
立ち絵を非同期トランジションしています。
```

trans=crossfade についてはよく使うので、簡単に「fade」属性を使うこともできます。属性値には *trans=crossfade* では time 属性に指定していたトランジションの時間を指定します。

■同期 / 非同期トランジションについて

テキスト表示と同時に別のこととする非同期動作は比較的重い（遅い）処理です。そもそもトランジションはそれ単体でも重い処理です。近頃のパソコンなら問題ありませんが、あまり同時かつ大量に使うと表示がカクカクしたりするかもしれません。

昔のゲームではパソコンの性能の問題でほぼ同期トランジションが使われていました。

非同期動作のいいところはプレイヤーを待たせないところです。同期トランジションでは画像が切り替わる間にテキストが進むまでプレイヤーを待たせてしまいます。クリックすれば飛ばせますがそれすら面倒な人もいます。稀にクリックしても飛ばせないゲームもありますが、立ち絵の表情が変わることに待たされるようなゲームはクソゲーです。内容に関係なく開始 30 秒でゴミ箱行きです。wt タグの canskip 属性は非常に紹介したくなかったのですが、表現の幅を狭めるのもよろしくないので紹介しました。使わないであげてください。切り替えにトランジションを使わない、というのもアリかもしれません。ここぞという場面でのみ使った方が効果的……かはわかりませんが。

現状、立ち絵の表情変更では非同期トランジション、背景ごと変わる場合は同期トランジション、という使い方がメジャーなように見えます。非同期トランジションにすると立ち絵なんか見なくなってしまう、という副作用もあるのですが、表情差分の種類は増えてる傾向にあります。いや最近は落ち着いているかもしれないです。増やしても誰が見ているんだろうと結構疑問だつたりします。ボイスを最後まで聞く人なら立ち絵にも目がいくんでしょうか。ボイスもボイスカットや多重再生などの機能についてどんどん飛ばせる方向になってますね。表情を見せるためか、テキストの横にフェイスウィンドウを出しているゲームもあります。

■不透明度

*start|スタート

@リビング 昼

@しおり ポーズ b 私服 驚く 前 中

背景とキャラクタを表示しました。

@しおり opacity=128

【しおり】「不透明度を半分にしました。」

@しおり opacity=0

【しおり】「不透明度を 0 にしました。」

@しおり opacity=255

【しおり】「不透明度を 255 に戻しました」

「opacity」属性を使うとレイヤの不透明度を 0 ~ 255 まで変更できます。不透明度が低いと画像が半透明になり、後ろのレイヤの画像が透けて見える用になります。

す。高いと不透明になるため後ろのレイヤの画像は見えません。デフォルトでは最大値の 255 になっているため、立ち絵の後ろの背景は完全に隠れています。`@しおり opacity=128` とすると不透明度が半分になるため後ろの背景が透けてみえるようになります。`@しおり opacity=0` とすると、完全に透明になり立ち絵が見えなくなってしまいます。`@しおり opacity=255` と不透明度を 255 に戻すとまた立ち絵が見えるようになります。`opacity` 属性は立ち絵だけでなく、背景、イベント絵、環境レイヤについても同じように使用できます。

■time 属性によるアクション

アクションを使うと、属性の値を連続的に変化させることができます。単純に属性に値を指定するだけでは瞬時に切り替わってしまいますが、アクションを使うと動きのある表現が実現できます。

```
*start|スタート
```

```
@リビング 昼
```

```
@しおり ポーズ a 私服 驚く 奥 左  
背景とキャラクタを表示しました。
```

```
@しおり xpos=200 time=1000
```

```
【しおり】「xposの値を 1 秒間で変更しています。」
```

```
@リビング xpos=-50 ypos=300 time=2000
```

```
背景の xpos と ypos の値を 2 秒間で変更しています。
```

```
@しおり opacity=0 time=3000
```

```
【しおり】「3 秒間で立ち絵を透明にしています。」
```

属性の値を変更する際に `time` 属性でミリ秒単位の時間を指定すると、属性の現在の値から指定された値まで、指定した時間かけて変化するようになります。これを「アクション」といいます。

`@しおり xpos=200 time=1000` では、`xpos` 属性の値が現在の 0 から指定された 200 まで 1 秒間で変化します。`xpos` 属性の値が「0,1,2,3,...,199,200」と変化していくので、見た目としては立ち絵が少しづつ右に移動していきます。`@リビング xpos=-50 ypos=300 time=2000` のように `xpos` 属性と `ypos` 属性を同時に指定すると、どちらも 2 秒間で変化します。`xpos`、`ypos` 属性だけでなく、`opacity` 属性についても同じように使用できます。

```
*start|スタート  
@リビング 昼  
@layer name=お星様 file=星 show  
背景と環境レイヤを表示しました。
```

```
@お星様 xpos=-200 time=1000 sync  
環境レイヤを同期アクションで移動しました。
```

```
@リビング xpos=100 ypos=-100 time=2000 sync  
背景を同期アクションで移動しました。
```

```
@お星様 opacity=0 time=3000 nosync  
環境レイヤを非同期アクションで透明にしています。
```

トランジションと異なり、アクションはデフォルトで非同期動作になっています。デフォルト値が違うだけで、動作の仕組みはトランジションの同期 / 非同期動作と同じです。同期動作の場合はアクションが終了するまでスクリプトの実行が中断されます。「sync」「nosync」属性でそれぞれ指定できます。非同期動作はページ送りでキャンセルされます。

```
*start|スタート  
@リビング 昼  
@layer name=お星様 file=星 xpos=-200 ypos=-200 show  
背景と環境レイヤを表示しました。
```

```
@お星様 xpos=200 ypos=200 time=10000 nowait  
環境レイヤを非同期アクションで移動しています。
```

nowait属性をつけているのでクリックしてもキャンセルされません。

```
@お星様 stopaction  
強制的にキャンセルしました。
```

「nowait」属性もトランジション同様に使えます。トランジションをキャンセルするのは「stoptrans」属性でしたが、アクションは「stopaction」属性でキャンセルできます。`@stopaction` とタグとして使うと全てのアクションをキャンセルすることもできます。

```
*start|スタート  
@街 夕  
@かえで ポーズ a 私服 無表情 奥  
背景と立ち絵を表示しました。
```

```
@街 xpos=-200 time=2000  
@かえで 右 time=2000  
背景と立ち絵を非同期アクションで移動しています。
```

```
@街 xpos=200 time=10000 nosync  
@かえで 左 time=10000 nosync  
@wact  
街と立ち絵を同期アクションで移動しました。
```

背景と立ち絵など、複数の画像に同時に非同期アクションを使う場合はそのままタグを並べれば OK です。トランジションにおける begintrans タグと endtrans タグのようなタグはありません。

同期アクションを同時に使う場合は「wact」タグが使えます。wt タグではトランジションの終了を待ちましたが、wact タグではアクションの終了を待てます。非同期アクションを同時に実行して、@wact でまとめてアクションが終了するのを待ちます。canskip 属性でスキップできないようにすることも可能です。

```
*start|スタート  
@街 昼 fade=1000 nosync  
@街 xpos=-200 nosync time=2000 nosync  
@wat  
背景のトランジションとアクションを同時に実行しました。
```

wt タグではトランジション、wact タグではアクションの終了を待ちますが、「wat」タグではトランジションとアクションの両方の終了を待ちます。canskip 属性についても同じく使用できます。

wt、wact、wat タグは、そのタグの時点でトランジションやアクションが実行されている場合に、その全てが終了するまでスクリプトの実行を停止します。canskip が省略されるか、*canskip=false* の場合にクリックされれば、実行されているトランジション / アクションを全てキャンセルします。

これらのタグの時点で 1 つも実行されていない場合は、そこで停止せずに単純に無視されます。

```
*start|スタート  
@街 昼 fade=500  
背景を表示しました。
```

```
@街 xpos=200 ypos=-200 opacity=0 time=2000  
背景を非同期移動しながら透明にしています。
```

```
@街 xpos=0 ypos=0 time=2000  
@街 opacity=255 time=4000  
背景を違う時間で非同期移動しながら不透明にしています。
```

xpos、ypos 属性と opacity 属性に対して同時にアクションを使う事もできます。@街 *xpos=200 ypos=-200 opacity=0 time=2000* のように 1 つのタグに書いた場合は time 属性による時間指定が共通になりますが、タグを 2 つにわければ異なる時間を指定することも可能です。ただし xpos と ypos については特別で、これらを別々にすることはできません。

```
*start|スタート  
@街 昼  
@layer name=星 1 file=星 show ypos=100  
@layer name=星 2 file=星 show ypos=0  
@layer name=星 3 file=星 show ypos=-100  
背景と環境レイヤを表示しました。
```

```
@星 1 xpos=300 time=2000 accel=accel  
@星 2 xpos=300 time=2000 accel=const  
@星 3 xpos=300 time=2000 accel=decel  
星 1 を加速移動、星 2 を等速移動、星 3 を減速移動しています。
```

```
@星 1 xpos=-300 time=4000 accel=acdec  
@星 2 xpos=-300 time=4000 accel=const  
@星 3 xpos=-300 time=4000 accel=accos  
星 1 を加減速移動、星 2 を等速移動、星 3 をコサイン加減速移動しています。
```

復習ですが ypos 属性は数値が大きいほど表示位置としては上なので、実行すると星 1 が一番上、星 2 が真ん中、星 3 が一番下に表示されます。

「accel」属性を使うことで、属性の値の変化を加速させたり減速させたりすることができます。アクション全体でかかる時間は time 属性に指定した値で変わりません。属性値に「accel」を指定すると最初はゆっくり、後から急激に値が変化するようになります。

なります。「decel」を指定すると最初は急激に、後からゆっくりと変化するようになります。「const」を指定すると「accel」属性を使っていない場合と変わらず、一定の速さで値が変化します。

「acdec」と「accos」は加減速で、前半は加速、後半は減速するようになります。実行してみるとわかりますがほとんど変わりません。「accos」の速度変化の方がわずかに緩やかになります。

■path 属性によるアクション

*start|スタート

@街 昼

@layer name=星 1 file=星 show

背景と環境レイヤを表示しました。

@星 1 path="(200,0,0),(200,200,255),(0,0,128)" time=4000

移動しています。

@星 1 path="(-200,0,255),(-200,200,255)" time=4000 spline

スプライン移動しています。

「path」属性を使うと複数点の移動と不透明度のアクションができます。属性値は「(xpos の値 ,ypos の値 ,opacity の値)」というセットをコンマでつないで、いくつか並べた形になります。`@星 1 path="(200,0,0),(200,200,255),(0,0,128)" time=4000` では、「`xpos=200,ypos=0,opacity=0`」「`xpos=200, ypos=200, oopacity=255`」、「`xpos=0, ypos=0, opacity=128`」が3つセットになっています。time 属性に指定した時間でその3点を順番に移動していきます。位置のみを変化させて、不透明度を変化させたくない場合は、`opacity` の部分を全て 255 など一定の値にしておきます。

path 属性とともに「spline」属性を使うとスプライン補間された経路で各点を移動します。各点をなめらかにつないで移動させることができます。他に sync、nosync、nowait 属性が使えますが、「accel」属性は使えません。



■画像の回転

*start|スタート

@街 昼

背景を表示しました。

@街 rotate=180

背景の角度を 180 度にしました。

@街 rotate=0

角度をもどしました。

「rotate」属性を使うと画像を回転できます。属性値には反時計回りで何度の角度に回転させるかを指定します。マイナスの値を指定すると時計回りの角度になります。背景だけでなく立ち絵やイベント絵、環境レイヤも同じく回転できます。

*start|スタート

@リビング 昼

背景を表示しました。

@リビング rotate=360 time=3000

背景を反時計回りに 1 回転させています。

@リビング rotate=-360 time=6000 sync accel=acdec

背景を時計回りに 2 回転させました。

time 属性によるアクションも使えます。`rotate=360 time=3000` では、アクションによってデフォルトの 0 から、指定された 360 まで「0,1,2,3...359,360」のように値が変化していくので、見た目としては反時計回りに 1 回転することになります。次の `rotate=-360 time=6000` では現在の値の 360 から指定された -360 まで「360,359,358,...,1,0,-1,...,-359,360」と `rotate` の値が変化していきます。見た目としては時計回りに 2 回転することになります。sync、nosync、accel 属性も問題なく使えます。

■回転原点

*start|スタート

@街 昼

背景を表示しました。

@街 rotate=360 time=1000

背景を反時計回りに1回転しています。

@街 afx=0 afy=0

回転原点を画像左上に変更しました。

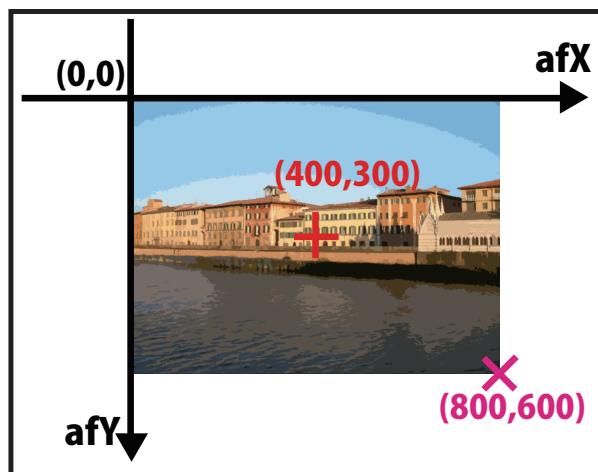
@街 rotate=720 time=1000

背景を反時計回りに1回転しています。

「afx」属性と「afy」属性で「回転原点」を指定できます。回転原点とは、回転の際に中心となる点(座標)で、その点の表示位置が回転の際にずれないように回転します。

回転原点は画像上の点で、画像左上を(0,0)、画像右下を(画像縦幅, 画像横幅)とした直交座標系で指定します。右の画像を見てください。背景画像のサイズは800×600です。横がX(afx), 縦がY(afy)で、afxは右がプラス、afyは下がプラスになります。前述の通り画像左上が(0,0)で、この背景画像のサイズは800×600のため画像右下の×の点が(800,600)になります。画像中心の+の点は(400,300)です。

背景のデフォルトの回転原点は画像中央になっています。800×600の画像では(400,300)です。なので初めの@街 rotate=360 time=1000では、右の画像の様に中央を中心として回転が実行されます。



画像中心が原点のとき



次に @街 *afx=0 afy=0* で回転原点を画像左上の (0,0) に変更しています。このとき表示位置が右下の方にずれてしまいますが、これについては後述します。とりあえず次の @街 *rotate=720 time=1000* では、右の画像のように画像左上を中心回転します。すでに 360 度まで回転しているので、もう 1 回転させるために 720 度を指定しています。

画像左上が原点のとき



*start|スタート

;回転が見やすいように位置を上にずらしています
@しおり 制服 ポーズ b 悲しみ 奥 中 ypos=300
立ち絵を表示しました

@しおり *rotate=180 time=2000*
立ち絵を反時計回りに 180度回転しています。

@しおり *afx=center afy=center*
回転原点を中心回転しました。

@しおり *rotate=0 time=2000*
立ち絵を時計回りに 180度回転しています。

立ち絵のデフォルトの回転原点は、画像中央ではなく画像下端中央になっています。なので回転は画像下端を中心にして実行されます。

また、*afx*、*afy* 属性では、数値で座標を指定する代わりにせずに文字で指定することもできます。*afx* 属性に「center」を指定すると、回転原点の X 座標が画像中央の点の X 座標に一致します。*afy* 属性に「center」を指定すると、回転原点の Y 座標が画像中央の点の Y 座標に一致します。よって、*afx=center afy=center* とすると画像中央が回転原点となります。また、*afx* 属性には「left」または「right」を指定することもできます。*afx=left* の場合は画像の左端、*afx=right* の場合は画像の右端が回転原点になります。*afy* 属性には「top」または「bottom」が指定でき、それぞれ画像上端、画像下端が回転原点になります。これでいうと、立ち絵のデフォルトの回転原点は

afx=center afy=bottom となります。

背景、イベント絵、環境レイヤのデフォルトの回転原点は画像中央、つまり *afx=center afy=center* になっています。立ち絵のみ異なっているのは、立ち絵の足元の位置が回転した際にずれないようにするためです。サンプルではそこまで描かれていませんが、足元まで含めて全身が描かれている立ち絵画像を使う場合、画像下端中奥はちょうど足元になります。立ち絵は一応地面に立っているはずなので接地面の足元がずれなければ都合のいいことが多いはずです。

■表示原点

*start|スタート

;回転が見やすいように位置を上にずらしています

@しおり 制服 ポーズ b 悲しみ 奥 中 ypos=300

立ち絵を表示しました

@しおり rotate=180 time=2000

立ち絵を反時計回りに 180度回転しています。

@しおり *afx=center afy=center*

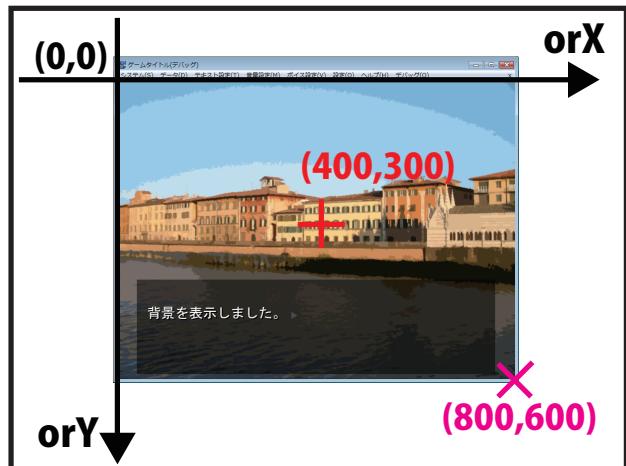
回転原点を中心にしました。

@しおり rotate=0 time=2000

立ち絵を時計回りに 180度回転しています。

回転原点の他に、「表示原点」があります。回転原点は回転の中心となる画像上の座標でしたが、表示原点は表示の中心となるウィンドウ上の座標になります。ウィンドウ左上が (0,0)、ウィンドウ右下が (ウィンドウ幅 , ウィンドウ高さ) となる直交座標系で指定されます。

右の画像を見てください。回転原点とほぼ同じですが、画像上ではなくウィンドウ上の座標であることに注意してください。ウィンドウサイズも 800×600 なのでウィンドウ右下は (800,600)、ウィンドウ中央は (400,300) になっています。



画像は表示原点を中心として表示されます。つまり、*xpos=0 ypos=0* のときに回転原点と表示原点が重なるように表示されます。

@街で背景を表示すると、回転原点(画像中央)がデフォルトの表示原点であるウィンドウ中央と重なる位置に表示されます。*xpos=100* とすればそこから右に 100px ずれ、*ypos=100* とすれば上に 100px ずれる、という仕組みになっています。

@街 *orx=0 ory=0* と表示原点をウィンドウ左上に変更すると、回転原点(画像中央)が表示原点(ウィンドウ左上)に重なるように画像の位置が変更されます。逆に回転原点が @街 *afx=left afy=top* と変更されても、それに伴って画像位置が変更されます。ここでは回転原点である画像左上が表示原点であるウィンドウ左上と重なるため、結果としては最初の表示と同じになります。

回転原点についても表示原点と同じように *orx* 属性は「left」「center」「top」(それぞれウィンドウ左端、中央、右端の X 座標)、*ory* 属性は「top」「center」「bottom」(それぞれウィンドウ上端、中央、下端の Y 座標)で指定できます。

また、回転原点はウィンドウ上の座標ではありますが、画像ごとに別々に設定されます。ウィンドウに 1 つしかない、という事ではありません。

回転原点は背景、立ち絵、イベント絵、環境レイヤのすべてでウィンドウ中央がデフォルトとなっています。

3 章で立ち絵を初めて表示したとき、立ち絵の位置が上にずれていきました。それは回転原点と表示原点の機能によって、画像下端が画面中央となるような位置になっていたためです。その位置ではまずいので、立ち絵については特別に縦位置のオフセット(*yoffset*)を指定できるようになっていた、ということになります。オフセットに指定されたピクセル数だけ表示原点の位置がずれたような動作になります。

■拡大率

*start|スタート

@街 昼

背景を表示しました。

@街 zoomx=50

背景の幅を 50%に縮小しました。

@街 zoomy=200

背景の高さを 200%に拡大しました。

@街 zoom=100

背景の拡大率を元に戻しました。

「zoomx」「zoomy」属性に画像の拡大率を指定することができます。それぞれ横方向、縦方向の拡大率で、デフォルトの 100が画像のサイズそのままとなっています。50を指定すると通常の半分、200を指定すると通常の倍、のようになっています。「zoom」属性を使って zoomx,zoomy属性に一括で同じ値を指定することもできます。

*start|スタート

@街 昼

@しおり 前 制服 ポーズ a 笑顔

背景と立ち絵を表示しました。

@街 zoom=50 time=2000 sync

背景を 50%に縮小しました。

@しおり zoom=200 time=2000

立ち絵を 200%に拡大しています。

zoom、zoomx、zoomy属性についても time属性によるアクションが使えます。rotate属性での回転と同様に、拡大縮小も回転原点を中心として実行されます。つまり、回転原点の表示位置は拡大、縮小されてもずれません。

*start|スタート

@街 昼

@しおり 前 制服 ポーズ b 無表情

背景と立ち絵を表示しました。

@街 zoom=-100 time=2000

背景を -100% に拡大しています。

@しおり zoomx=-100 time=2000

立ち絵の幅を -100% に拡大しています。

@街 zoom=-200 time=2000

背景を -200% に拡大しています。

zoom、zoomx、zoomy 属性にマイナスの値を指定すると回転原点を中心として反転します。「zoomx」の場合は左右に反転、「zoomy」の場合は上下に反転、「zoom」なら上下左右が反転した見た目になります。-200 を指定すれば反転した上で 2 倍に拡大されます。

■反転

*start|スタート

@街 夕

@しおり 奥 私服 ポーズ b 無表情

背景と立ち絵を表示しました。

@街 flipy

背景を縦方向に反転しました。

@しおり flipx

立ち絵を横方向に反転しました。

@街 flipx

背景を横方向にも反転しました。

@街 flipx=false flipy=false

背景の反転を戻しました。

「flipx」「flipy」属性を使って画像を反転することもできます。true を指定すると反転、false を指定すると反転を元に戻します。属性値の true は省略できることを考えているでしょうか。flipx、flipy による反転は回転原点に関係なく、画像の表示位置

がずれないように反転されます。手軽に使えるので反転する場合は「zoom」より便利かもしれません。

■傾斜率

*start|スタート

@街 昼

@しおり 前 制服 ポーズ b 無表情
背景と立ち絵を表示しました。

@街 slantx=100

背景を横方向 100に傾斜しました。

@しおり slantx=-100

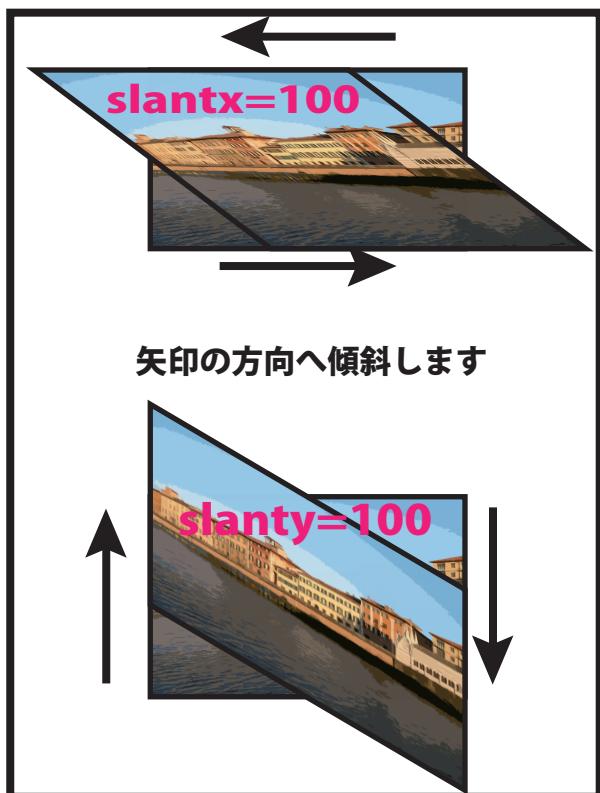
立ち絵を横方向 -100に傾斜しました。

@街 slantx=0 slanty=100

背景を縦方向 100に傾斜しました。

「slantx」「slanty」属性を使うと画像を傾斜させることができます。どちらも 0 がデフォルトで全く傾斜していない状態です。

傾斜させると四角形が平行四辺形となるように変形されます。100 のとき鋭角 45 度の平行四辺形になります。言葉で上手く説明できないので右の画像を見てください。値が大きいほど傾斜がきつくなります。マイナスの値では逆方向に傾斜します。傾斜も回転拡縮と同じく、回転原点が中心となり、その位置がずれないよう傾斜されます。



矢印の方向へ傾斜します

*start|スタート

@街 昼

背景を表示しました。

@街 slantx=100 time=2000

背景を横方向 100に傾斜しています。

@街 slantx=0 time=2000 sync

背景の傾斜をもどしました。

@街 slanty=100 time=2000

背景を縦方向 -100に傾斜しています。

傾斜でも time 属性によるアクションが使えます。注意点も特にありません。「slantx」と「slanty」を同時にアクションで使うこともできます。上のスクリプトではよくわからなくなるので1つずつ使っています。

■ぼかし

*start|スタート

@街 昼

背景を表示しました。

@街 slantx=100 time=2000

背景を横方向 100に傾斜しています。

@街 slantx=0 time=2000 sync

背景の傾斜をもどしました。

@街 slanty=100 time=2000

背景を縦方向 -100に傾斜しています。

「blurx」「blury」属性を使うと画像をぼかすことができます。どちらも 0 がデフォルトで全くぼかしをかけていない状態です。blurx 属性が横方向、blury 属性が縦方向のぼかしです。画像が変形されるわけではないので回転原点は関係ありません。「blur」属性で blurx、blury 属性に同じ値に指定することもできます。

*start|スタート

@しおり 前私服 ポーズ a 笑顔

立ち絵を表示しました。

@しおり blurx=50 time=1000

立ち絵を横方向にぼかしています。

@しおり blurx=0 blury=50 time=1000 accel=accel

立ち絵を縦方向にぼかしました。

@しおり blur=0 time=1000 sync

ぼかしを消しました。

@しおり allclear

全クリアしました。

ぼかしでも time 属性によるアクションも使えます。

大きな値でぼかしを使うと、ぼかしを 0 にしても周りに色が残ってしまうことがあります。上のサンプルでもぼかしを消した時点で周りに色が残ってしまいます。ぼかしに限らず回転や拡縮、移動などでも急激なアクションを使うと残ってしまったりします。これが起こるのは、もともとの画像サイズよりも広い範囲に色が描画された場合です。ぼかして大きな値を入れると本来の画像のサイズ外にも色が塗られるので、その部分が残ってしまいます。

回避策としては、画面外にはみ出すような大きな値を使わないか、画像サイズ自体を大きくするかです。画像の周りに透明な部分を付け足せば画像サイズは大きくできます。

「allclear」属性を使うと画像サイズの外の部分に残っている色も強制的に綺麗にしてくれます。すぐに「allclear」を使うことができる場合はこの属性を使えばあまり目立たなくできるかもしれません。上のスクリプトでは、ぼかしを消してからプレイヤーが次のテキストに進んで「@ しおり allclear」が実行されるまで色が残ったままになってしまいます。

■ラスタースクロール

*start|スタート

@街 昼

@しおり 奥 私服 ポーズ a 笑顔

背景と立ち絵を表示しました。

@街 raster=100

背景をラスタースクロールしています

@街 raster=0

停止しました。

@街 raster=100 rasterlines=200 rastercycle=2000

ラスタースクロールしています。

@街 raster=200

スクロールする横幅を変更しました。

@街 raster=200 rasterlines=400

ラインの縦幅を変更しました。

@街 raster=200 rastercycle=10000

ラスターの時間を変更しました。

「raster」属性では横ラスタースクロールが使えます。画像が横方向にグニャグニヤとスクロールします。

raster 属性には、横方向に最大でどれだけスクロールさせるかをピクセル単位で指定します。raster=0 とすればラスタースクロールを停止できます。

「rasterlines」属性には縦方向のラインの幅をピクセル単位で指定します。この属性に指定したピクセル数ごとに左右どちらにスクロールされるかが変わります。デフォルト値は 100 になっています。「rastercycle」属性には左右 1 往復スクロールするのにかかる時間をミリ秒単位で指定します。デフォルト値は 1000 です。

rasterlines 属性または rastercycle 属性を指定する場合は必ず「raster」も同時に指定する必要があります。@街 rasterlines=400 ではなく @街 raster=200 rasterlines=400 とする必要があります。

```
*start|スタート
```

```
@街 昼
```

```
@しおり 奥 私服 ポーズ a 笑顔
```

```
背景と立ち絵を表示しました。
```

```
@街 raster=100 time=2000
```

```
背景をラスタースクロールしています。
```

```
@街 raster=0 time=2000
```

```
停止しています。
```

raster 属性については time 属性によるうアクションが使えます。`@街 raster=100 time=2000` では raster 属性の値が「0,1,2,...,99,100」と変化していくので徐々にスクロールされる横幅が広くなっていきます。`@街 raster=0 time=2000` ではその逆で、2 秒かけてラスタースクロールが停止します。

rasterline 属性と rastercycle ゾック製にはアクションは使えません。time 属性に関係なく、すぐに指定された値に変更されます。

ここまで、サンプルスクリプトでは背景と立ち絵くらいしか使っていませんが、回転、拡大率、反転、傾斜度、ぼかし、ラスタースクロールは背景、立ち絵、イベント絵、環境レイヤで同じように使えます。

■相対指定

```
*start|スタート
```

```
@街 昼
```

```
@しおり 奥 私服 ポーズ a 笑顔
```

```
背景と立ち絵を表示しました。
```

```
@しおり xpos=@100
```

```
【しおり】「右に 100px 移動しました。」
```

```
@しおり xpos=@100
```

```
【しおり】「もう 1 度右に 100px 移動しました」
```

```
@しおり xpos=@-400
```

```
【しおり】「左に 400px 移動しました。」
```

属性値の先頭に @ (半角アットマーク) をつけると、現在の値からの相対指定になります。その属性の現在の値に、指定した値を足した値が実際の値になります。

xpos はデフォルトで 0 なので、最初の @しおり xpos=@100 では xpos の値が $0+100=100$ になります。次の @しおり xpos=@100 では、この時点で xpos の値は 100 になっているので $100+100=200$ になります。最後の @しおり xpos=@-400 では、 $200+(-400)=-200$ になります。

相対指定は time 属性によるアクションが使える属性であれば使用できます。ここで time 属性によるアクションが使える属性をまとめておくと、「xpos」「ypos」「opacity」「rotate」「zoomx」「zoomy」「zoom」「slantx」「slanty」「blurx」「blur」「blur」「raster」となります。

■パーセント指定

*start|スタート

@街 昼

@しおり 奥 私服 ポーズ a 笑顔

背景と立ち絵を表示しました。

@しおり xpos=100%

【しおり】「右 100%の位置に移動しました。」

@しおり xpos=0% ypos=50%

【しおり】「上 50%の位置に移動しました。」

@しおり rotate=100% time=2000

【しおり】「反時計回りに一回転しています。」

@しおり opacity=50%

【しおり】「不透明度を半分にしました。」

属性値の末尾に %(半角パーセント) をつけるとパーセント指定になります。パーセント指定が使える属性は相対指定が使える属性と同じです。

zoomx、zoomy、zoom、slantx、slanty 属性については元から率による指定なので % をつけてもつけなくても変わりません。

ピクセル単位の指定である xpos、ypos、blurx、blur、blur、raster については、0% が 0、100% が x では画面横幅、y では画面縦幅の半分のピクセル数になります。デフォルトの画面横幅は 800 なので @しおり xpos=100% では 800 の半分の 400 だけ右にずれます。次の ypos=50% では、デフォルト画面縦幅 600 の半分の 300 の 50% で、150 だけ上にずれることになります。xpos=-50% のようにマイナスをつけて左にずら

すこともできます。

rotate 属性では 1 回転の 360 が 100% に対応します。oapcity 属性では最大値の 255 が 100% になります。

■初期値指定

*start|スタート

@街 昼

@しおり 奥 私服 ポーズ a 笑顔

背景と立ち絵を表示しました。

@しおり xpos=-100:100 time=2000

【しおり】「xpos=100から xpos=-100に移動しています。」

@しおり zoom=100:200 time=2000

【しおり】「200%から 100%に拡大しています。」

@しおり pos=中:右外 time=2000

;@しおり 中:右外 time=2000 でも OK

【しおり】「右外から中に移動しています。」

time 属性によるアクションでは、現在の値から指定された値まで属性が変化します。属性値の後ろに「: 初期値」の形でくっつけて、アクションを始める最初の値を指定することができます。

@しおり xpos=-100:100 time=2000 とすると、xpos 属性が「100,99,98,...,-99,-100」のように変化することになります。次の @しおり zoom=100:200 time=2000 でも同様に「200,199,198,...,101,100」と変化していきます。立ち絵の pos=中:右外については属性名を省略して中:右外のように使うこともできます。

初期値指定が使える属性は、初期値指定と同じく time 属性によるアクションが使える属性全てとなります。

相対指定、パーセント指定、初期値指定を組み合わせて使うこともできます。
xpos=100:@-100 や xpos=@-40%:@20 のように z なりますが、必要になったことも無いので特に説明しません。

■回転原点のアクション

*start|スタート

@街 昼

@しおり 奥 私服 ポーズ a 笑顔

背景と立ち絵を表示しました。

@街 *afx=0 afy=0 time=2000*

背景の回転原点を変更しています。

@街 *afx=center afy=center rotate=180 time=2000*

回転原点を変更しながら回転しています。

回転原点についてもアクションが使えます。*@街 afx=0 afy=0 time=2000* では、回転原点の位置が変わるにつれて画像の表示位置も変わります。*@街 afx=0 afy=0 rotate=180 time=2000* ではそれに加えて画像の回転も行われます。

他のアクションをえる属性と異なり、*afx*、*afy* 属性のアクションでは相対指定、パーセント指定、初期値指定は使えません。

回転原点まで使うとそこそこの複雑な動きもできるようになりますが、実際に複雑な動きをさせる場合はアクションを定義して使うことが多いです。アクション定義の仕方は後ほど紹介します。

■単語登録

余白を埋めるためにまた脇道にそれますが、IME の単語登録はなかなか便利です。何それ？という人は検索してみてください。

*@しおり*なんて毎回真面目に打つのは結構だるいです。今は *q s* で *@しおり*、*q k* で *@かえで*と変換できるようになっています。*q s*、*q k* に特に意味はありません。タイトルごとによく使う文字列はどんどん登録してしまっています。

共通で使っているものとしては、例えば *s* の変換が半角スペースになっています。属性の間のスペースを打つためにいちいち半角 / 全角切り替えなんてやめましょう。これは 5 章のボイスの解説で出てきますが、*p d |* で *delayrun=ラベル0* に変換できます。よく使うのに属性名が長くて大変なんですね。

私のやり方を真似する必要は無いのですが、まあこんな省力化の方法もあるので参考にしてください。

■カメラ操作

KAGEX には「カメラ」という概念があります。「stage」「char」「event」「layer」タグによって表示される画像は、仮想的なカメラを通して画面にうつされている、と考えます。

カメラ機能は一応紹介しますが、使い方が面倒なので実際はそれほど使われなかりります。

```
*start|スタート
```

```
@街 昼
```

```
@しおり 前 右 制服 ポーズ a 笑顔
```

```
背景と立ち絵を表示しました。
```

```
@env camerax=50
```

```
カメラ位置を右に 50ずらしました。
```

```
@env cameray=50
```

```
カメラ位置を上に 50ずらしました。
```

```
@env camerazoom=200
```

```
カメラの拡大率を 200%にしました。
```

```
@env cameralrotate=45
```

```
カメラを反時計回りに 45度傾けました。
```

カメラは「env」タグを使って操作します。「env」タグはカメラ以外にもゲーム全体の操作をする場合に使いますが、それは後ほど解説します。

env タグの「camerax」「cameray」属性でカメラの位置をずらします。デフォルトではどちらも 0 になっています。カメラを右に移動させると、それに映されている画面全体は左にずれ、カメラを上に移動させるとそれに映されている画面全体は下にずれます。「camerazoom」属性を使うとカメラの拡大率が指定できます。デフォルトは 100 で、*camerazoom=200* であれば画面全体が 2 倍に拡大されます。「cameralrotate」属性を使うとカメラを回転させることができます。指定する値の意味は「rotate」属性と同じです。

```
*start|スタート  
@街 昼  
@しおり 前 右 制服 ポーズ a 笑顔  
背景と立ち絵を表示しました。
```

```
@env camerax=-150 time=2000  
カメラ位置を左に 150ずらしています。
```

```
@env cameray=150 time=2000  
カメラ位置を上に 150ずらしています。
```

```
@env camerazoom=200 time=2000  
カメラの拡大率を 200%にしています。
```

```
@env camerarotate=180 time=2000  
カメラを反時計回りに 180度傾けています。
```

time 属性によるアクションも使えます。アクションさせた方がカメラの動きがわかりやすいかもしれません。

カメラは単純に画面全体を動かすのにも使えますが、画像の Z 位置を指定すると 2.5 次元的な動きが実現できます。2.5 次元というのは、画像を奥行き方向に配置できるという意味で使っています。これまでの横方向 (X)、縦方向 (Y) に加えて、画像には奥行き方向 (Z) が存在します。デフォルトでは 100 で、数字が大きいほど手前側となります。3 次元 (3D) ではないのでカメラが回りこんだりする事はできませんが、いくらか 3 次元っぽく 2 次元の画像が動かせます。

```
// ■ levelz  
// 立ち絵の Z 位置を表示レベルごとに指定します。  
levelz[0] = 100;  
levelz[1] = 100;  
levelz[2] = 100;  
  
// ■ bglevelz  
// 背景の Z 位置を指定します。  
bglevelz = 100;
```

背景と立ち絵の Z 位置はあらかじめ定義しておく必要があります。「init」フォルダの envinit.otjs を開いて上のような部分を探してください。立ち絵の Z 位置は表示レ

ベルごとに「`levelz[表示レベル] = Z位置;`」の形で指定します。背景のZ位置は「`bglevelz = Z位置;`」の形で、すべての背景共通に指定します。Z位置は0より大きい数字である必要があります。ここでは以下のようにしてください。

```
// ■ levelz
// 立ち絵のZ位置を表示レベルごとに指定します。
levelz[0] = 100;
levelz[1] = 150;
levelz[2] = 100;

// ■ bglevelz
// 背景のZ位置を指定します。
bglevelz = 50;
```

表示レベル2はサンプルスクリプトでは使用していないので100のままで。表示レベル1のときの立ち絵は、表示レベル0の画像を1.5倍に拡大した画像なので、表示レベル1のときのZ位置も表示レベル0の時の位置の1.5倍手前にしています。本当は距離と大きさの関係はこれほど単純ではないです。が面倒なのでこうしておきます。

背景はテキトーですが表示レベル1の立ち絵よりも2倍遠くにしています。カメラを使用する場合は、背景を準備する際にカメラからの距離も考える必要があります。場合によっては距離ごとにパーツ分けなどが必要かもしれません。カメラをちゃんと使おうとすると大変です。背景については、同人でフリー素材を使っているとアイレベルもバラバラだったり立ち絵が浮遊していたりするので、細かいことは気にしないというのもアリかもしれません。

`envinit.otjs`を上の通り変更して保存したら、toolsフォルダの`autosetting.bat`を実行して変更を適用してください。

Z位置は直接的にカメラとの距離を指定するのではなく、画像間の相対的な位置として指定します。Z位置のデフォルトは100で、この値を基準とします。Z位置が100のとき、カメラを左に10移動すると画像は右に10ピクセル移動します。Z位置が150であれば、右に15ピクセル移動し、50であれば右に5ピクセル移動します。

車に乗って外を見ると手前の景色が奥の景色よりも早く流れていきます。手前のものが奥の物よりも大きく移動しているように見える、ということです。カメラが移動するというのは見ている側の車の位置が移動するのと同じです。`xpos`、`ypos`属性で画像を移動するというのは、車は停止して、車の外の人（見られている側=被写体）

が歩いて移動しているのと同じです。

xpos、ypos 属性による位置指定も影響を受けます。おなじ `xpos=100` でも Z 位置が 100 の時は右に 100 ピクセル、Z 位置が 150 のときは右に 150 ピクセルずれるようになります。

```
*start|スタート  
@街 昼  
@かえで 奥 左 私服 ポーズ a 無表情  
@しおり 前 左 中 制服 ポーズ a 笑顔  
背景と立ち絵を表示しました。
```

```
@env camerax=-200 time=2000  
カメラ位置を左に 200ずらしています。
```

```
@env cameray=200 time=2000  
カメラ位置を上に 200ずらしています。
```

```
@env camerazoom=200 time=2000  
カメラの拡大率を 200%にしています。
```

```
@env camerarotate=180 time=2000  
カメラを反時計回りに 180度傾けています。
```

上のスクリプトを実行すると Z 位置の効果がわかると思います。

先ほど定義したとおり、背景の「街」の Z 位置は 50、表示レベル 0(奥)の「かえで」の Z 位置は 100、表示レベル 1(前)の「しおり」の Z 位置は 150 となっています。なので、カメラが左に 150 ずらしたとき、「街」は 100 ピクセル、「かえで」は 200 ピクセル、「しおり」は 300 ピクセルずつ右にずれます。結果としてどのように見えるかは実際に実行してみてください。

`camerazoom` 属性による拡大と `camerarotate` 属性による回転は Z 位置の影響を受けません。先ほどの車の例で言えば、車のなかで 5 倍双眼鏡を使うと、距離に関係なく全てのものが元の大きさの 5 倍に拡大して見えます。首を 45 度傾けても距離に関係なくすべてのものが 45 度傾きます。

```
*start|スタート  
@街 昼  
@かえで 奥 左 私服 ポーズ a 無表情  
@しおり 前 左中 制服 ポーズ a 笑顔  
@layer name=星 1 file=星 levelz=80 xpos=-50 ypos=300  
背景、立ち絵、星を表示しました。
```

```
@env camerax=-200 time=2000  
カメラ位置を左に 200ずらしています。  
  
@env cameray=200 time=2000  
カメラ位置を上に 200ずらしています。
```

環境レイヤの Z 位置は「levelz」属性によって指定できます。envinit.otjs で指定した 100 を基準とする位置指定と同じものです。

```
*start|スタート  
@街 昼  
@かえで 奥 左 私服 ポーズ a 無表情  
@しおり 前 左中 制服 ポーズ a 笑顔  
@layer name=星 1 file=星 xpos=-270 ypos=250 zoom=80 nocamera  
背景と立ち絵を表示しました。
```

```
@env camerax=-200 cameray=200 time=2000  
カメラ位置を左上に 200ずらしています。
```

```
@env camerazoom=200 cameralrotate=180 time=2000  
カメラの拡大率を 200%、180度回転しています。
```

char、stage、layer タグでは、「nocamera」属性を使うことができます。この属性に true が指定された画像は camerax、cameray、cameralrotate、camerazoom 属性の影響を受けなくなります。上のスクリプトで「星 1」の位置は全く動きません。画面上に日付や何かのパラメータ画像を表示する際などに使えます。

イベント絵については初めからカメラの影響を一切受けません。なので levelz 属性などで Z 位置を指定することもできません。

```
*start|スタート  
@街 昼  
@かえで 奥 左 私服 ポーズ a 無表情  
@しおり 前 左 中 制服 ポーズ a 笑顔  
背景と立ち絵を表示しました。
```

```
@env shiftx=-100 time=1000  
カメラを左に 100シフトしています。
```

```
@env shifty=100 time=1000  
カメラを上に 100シフトしています。
```

「shiftx」「shifty」属性は camerax、cameray 属性とほぼ同じですが、画像の Z 位置とは無関係に、指定したピクセル数だけ全体が移動します。

camerax=-100 では画像の Z 位置によって実際にずらされるピクセル数が変わりますが、*shiftx=-100* ではすべての画像が右に 100 ピクセルずれます。*shifty=100* では全ての画像が下に 100 ピクセルずれます。その他の点は camerax、cameray 属性と同じです。

```
*start|スタート  
@街 昼  
@かえで 奥 左 私服 ポーズ a 無表情  
@しおり 前 左 中 制服 ポーズ a 笑顔  
@layer name=星 1 file=星 xpos=-270 ypos=250 zoom=80 noshift  
背景、立ち絵、星を表示しました。
```

```
@env shiftx=-100 time=1000  
カメラを左に 100シフトしています。
```

```
@env shifty=100 time=1000  
カメラを上に 100シフトしています。
```

nocamera 属性と似ていますが、「*noshift*」属性もあります。この属性に *true* を指定すると、その画像は *shiftx*、*shifty* 属性の影響を受けなくなります。上のスクリプトでは「星 1」は全く動きません。使い所は *nocamera* 属性とそう変わりません。

イベント絵は、*shiftx*、*shifty* 属性についても影響を受けないようになっています。

```
*start|スタート  
@街 昼 blur=30  
@かえで 奥 左 私服 ポーズ a 無表情 blur=15  
@しおり 前 中 制服 ポーズ a 笑顔  
@layer name=星 1 file=星 xpos=-270 ypos=250 zoom=80 noshift blur=20  
背景、立ち絵、星を表示しました。
```

```
@街 blur=15 time=1000  
@かえで blur=0 time=1000  
@星 1 blur=5 time=1000  
@しおり blur=-15 time=1000  
かえでにピントを合わせています。
```

```
@街 blur=10 time=1000  
@かえで blur=-5 time=1000  
@星 1 blur=0 time=1000  
@しおり blur=-20 time=1000  
星にピントを合わせています。
```

KAGEX のカメラには、ここまで紹介してきたような簡易的な Z 位置による XY 位置の補正の機能しかありません。使うのに苦労する割に機能としては微妙かもしれません。被写界深度やピンぼけなどを表現する機能はありません。上のスクリプトのように blur 属性でぼかすだけでもそれっぽくは見えます。

■画面揺らし

```
*start|スタート  
@街 昼  
@しおり 前 右 制服 ポーズ a 笑顔  
背景と立ち絵を表示しました。
```

```
@quake time=1000 vmax=20 hmax=10  
画面を 1 秒間揺らしています。
```

```
@quake time=2000 sync  
画面を 2 秒間揺らしました。
```

「quake」タグを使うと画面全体を揺らすことができます。アクションなどでも揺らせますが quake タグはお手軽に使って便利だったりします。「vmax」属性は縦、「hmax」属性は横に、最大で何ピクセルだけ揺らすかを指定します。これらの属性に指定した範囲内でランダムに画面が揺らされます。

vmax、hmax 属性は省略するとデフォルト値の 10 になります。「time」属性には何ミリ秒の間画面を揺らすかを指定します。デフォルトでは非同期動作ですが、「sync」属性で同期動作にすることもできます。

```
*start|スタート  
@街 昼  
@しおり 前 右 制服 ポーズ a 笑顔  
背景と立ち絵を表示しました。
```

```
@quake  
画面を揺らしています。
```

quake タグの揺れはページ送りでキャンセルされません。

```
@stopquake  
画面揺らしを停止しました。
```

quake タグの time 属性を省略すると、いつまでも画面が揺れ続けることになります。また、quake タグの非同期動作はアクションやトランジションなどと異なり、ページ送りでキャンセルされません。揺れを停止するには「stopquake」タグを使う必要があります。

```
*start|スタート  
@街 昼  
@しおり 前 右 制服 ポーズ a 笑顔  
背景と立ち絵を表示しました。
```

```
@quake message  
画面を揺らしています。
```

ゆれゆれゆれ。

```
@stopquake  
揺れを停止しました。
```

「message」属性を使うと、メッセージ部分も画面と一緒に揺らすことができます。読みづらくなるのでテキスト表示と一緒に使用するのはお勧めしません。

■拡張トランジション

トランジションの種類として「crossfade」「universal」「scroll」の 3 つを紹介しま

したが、吉里吉里プラグインを使用するとその以外のトランジションが使用できます。

吉里吉里プラグインとは、吉里吉里の機能を拡張するためのプログラムで、開発用フォルダの「plugin」フォルダに入っている拡張子が dll のファイルがそれです。そこに入っているプラグインは KAGEX の動作に必要なのでゲームが完成した暁にはゲーム本体と共にそれらも配布することになります。プラグインを使うにはプラグインを吉里吉里に「リンク」するのですが、それら必須のプラグインのリンクは自動的に行われています。拡張トランジションの機能を含むプラグインは必須ではないためそこには含まれていません。

まずはプラグインファイルをコピーしてきます。SDK フォルダの「プラグイン」フォルダの「extrans」フォルダに入っている「extrans.dll」を、開発用フォルダの「plugin」フォルダにコピーしてください。

```
// 追加のプラグインのリンク  
// 例 : Plugins.link("saveStruct.dll");
```

次に吉里吉里にリンクします。プロジェクトフォルダの「init」フォルダの「plugins.tjs」を開くと、上のように書かれています。プラグインのリンクは「*Plugins.link("プラグインの名前");*」のような形式になります。

```
// 追加のプラグインのリンク  
// 例 : Plugins.link("saveStruct.dll");  
Plugins.link("extrans.dll");
```

Plugins.link("extrans.dll"); という行を追加したら保存してください。「extrans.dll」が吉里吉里起動時にリンクされるようになります。これで準備は整ったので拡張プラグインを使用してみましょう。

拡張プラグインは KAGEX には直接関係ないので簡単に紹介するうだけに留めておきます。詳しくは吉里吉里リファレンス (<http://devdoc.kikyou.info/tvp/docs/kr2doc/contents/>) のトランジションのページ (<http://devdoc.kikyou.info/tvp/docs/kr2doc/contents/Transition.html#id295>) を参照してください。「拡張トランジションプラグイン」を読みながら属性値を適当にいじっていればわかってきます。

*start|スタート

@街 昼

背景を表示しました。

@街 夕 trans=wave time=10000 wavetype=2 maxh=50 maxomega=0.2

波トランジションをしました。

@街 夜 trans=mosaic time=2000 maxsize=20

モザイクトランジションです。

@向日葵 昼 trans=turn time=2000

ターントランジションです。

@向日葵 夕 trans=rotatezoom time=2000 factor=0 accel=0 twist=3 twistaccel=0

回転ズームトランジションです。

@街 昼 trans=rotatevanish time=2000 accel=2 twist=-2 twistaccel=0

回転消滅トランジションです。

@街 夕 trans=rotateswap time=2000 twist=2

回転入れ替えトランジションです。

@街 夜 trans=ripple time=2000 rwidth=64 roundness=1.0 speed=6.0 maxdrift=20

波紋トランジションです。

trans 属性にトランジションの種類、time 属性にトランジションの時間、それに加えて種類ごとの属性を指定します。詳細はリファレンスを参照してください。リファレンスにはありませんが sync、nosync 属性も使えます。

■トランジションの定義

よく使うトランジションをあらかじめ定義しておくことができます。開発用フォルダの tools フォルダの「register_trans.exe」を実行して、プロジェクトフォルダを選択してください。

「追加」ボタンをクリックすると新しくトランジションを定義できます。追加された「newtrans」を選択して、追加ボタンの下のテキストボックスに名前を入力、エンターキーでトランジションの名前を変更できます。ここでは「右スクロール」という名前にしておきます。「削除」ボタンをクリックで追加したトランジションを削除

することもできます。

左側のリストからトランジションを選択し、右側でそのトランジションの属性を設定します。「右スクロール」の属性は「トランジションの種類」を「Scroll」、「time」を「1000」、「from」を「left」、「stay」を「nostay」としてください。

それができたら、メニューの「出力」をクリックします。これを忘れる変更が反映されません。ここまで済めばスクリプト中で使えるようになります。

```
*start|スタート
```

```
@街 昼
```

```
背景を表示しました。
```

```
@向日葵 trans=右スクロール
```

```
右スクロールで背景を変更しました。
```

```
@街 夕 trans=右スクロール time=3000
```

```
time属性を上書きして右スクロールしました。
```

```
@街 夜 右スクロール stay=stayfore
```

```
stay属性を上書きして右スクロールしました。
```

trans 属性に定義したトランジション名を指定すると、そのトランジションが使用できます。それと同時にトランジションの属性を指定することで、定義された内容の一部分を変更してトランジションできます。`@街 夕 trans=右スクロール time=3000` では、time 属性のみ変更され、定義されている 1 秒ではなく 3 秒でトランジションが行われるようになります。

`@街 夜 右スクロール` のように、trans 属性の属性名は省略して、定義した名前のみで使うこともできます。基本的にこちらを使うことになるとおもいます。

■自動トランジションの定義

定義したトランジションを、画像変更時に自動的に使用するように定義しておくことができます。自動的に実行されるトランジションを「自動トランジション」といいます。

自動トランジションを定義する前に、使用するトランジションを定義しておきます。

先ほど使用した `register_trans.exe` をもう 1 度起動してください。「背景トランジ

ション」、「立ち絵トランジション」という2つのトランジションを追加しいて、トランジションの種類はどちらも「Crossfade」、「time」属性は背景トランジションが1000、立ち絵トランジションが500となるように設定します。立ち絵トランジションの方は「nosync」属性のチェックも入れてください。非同期トランジションになります。

以上が設定できたら「出力」メニューをクリックするのを忘れないでください。

次に自動トランジションを定義します。

```
// ■ envTrans  
// 舞台、時間、イベント絵のいずれかが変更された際の自動トランジション  
// を指定します。  
autoTrans["envTrans"] = "";  
  
// ■ stageTrans  
// 舞台または時間が変更された際の自動トランジションを指定します。  
// envTransよりも優先されます。  
autoTrans["stageTrans"] = "";  
  
// ■ timeTrans  
// 時間が変更された際の自動トランジションを指定します。  
// envTrans、stageTransよりも優先されます。  
autoTrans["timeTrans"] = "";  
  
// ■ eventTrans  
// イベント絵が変更された際の自動トランジションを指定します。  
// envTransよりも優先されます。  
autoTrans["eventTrans"] = "";  
  
// ■ charTrans  
// 立ち絵のポーズ、服装、表情のいずれかが変更された際の自動トランジ  
// ションを指定します。  
autoTrans["charTrans"] = "";  
  
// ■ poseTrans  
// 立ち絵のポーズが変更された際の自動トランジションを指定します。  
// charTransよりも優先されます。
```

```

autoTrans["poseTrans"] = "";

// ■ dressTrans
// 立ち絵の服装が変更された際の自動トランジションを指定します。
// charTransよりも優先されます。
autoTrans["dressTrans"] = "";

// ■ faceTrance
// 立ち絵の表情が変更された際の自動トランジションを指定します。
// charTransよりも優先されます。
autoTrans["faceTrans"] = "";

// ■ positionTrans
// 立ち絵の x/ypos、表示状態、表示レベルのいずれかが変更された際の自動
// トランジションを指定します。
autoTrans["positionTrans"] = "";

// ■ charDispTrans
// 立ち絵が表示または非表示にされた際の自動トランジションを指定します。
// positionTransよりも優先されます。
autoTrans["charDispTrans"] = "";

```

init フォルダの envinit.otjs を開いて上のような部分を探してください。「*autoTrans["自動トランジションの種類"] = "トランジション名";*」の形で定義します。「envTrans」など自動トランジションの種類ごとに、それぞれ使われるタイミングが異なります。envTrans に指定したトランジションは舞台、時間、イベント絵のいずれかが変更された際に使われます。その他の自動トランジションは上のスクリプト中の説明を読んでください。

同じ変更に対して複数の自動トランジションが定義できる場合、適用範囲の狭い自動トランジションが優先されます。たとえば、時間が変更された際の自動トランジションは「timeTrans」「stageTrans」「envTrans」の 3つに定義できますが、timeTrans は時間のみ、stageTrans は時間と舞台、envTrans は時間と舞台とイベント絵に適用されるので、この順番に優先されます。timeTransが指定されていれば timeTrans、timeTransが指定されていなければ stageTrans、stageTransも指定されていなければ envTrans、が時間が変更された場合の自動トランジションとして使われることになります。envTransも指定されていなければ自動トランジションは行われません。

```

// ■ envTrans
// 舞台、時間、イベント絵のいずれかが変更された際の自動トランジション
// を指定します。
autoTrans["envTrans"] = "背景トランジション";

// ■ charTrans
// 立ち絵のポーズ、服装、表情のいずれかが変更された際の自動トランジ
// ションを指定します。
autoTrans["charTrans"] = "立ち絵トランジション";

// ■ positionTrans
// 立ち絵の x/ypos、表示状態、表示レベルのいずれかが変更された際の自動
// トランジションを指定します。
autoTrans["positionTrans"] = "立ち絵トランジション";

```

長いので該当部分のみですが、サンプルとして上の 3 つを設定しておきます。これ以外は変更せず未指定のままです。指定ができたら autosetting.bat を実行しておきます。

```

*start|スタート
@街 昼
; 背景トランジションが実行されます
背景を表示しました。

@しおり 前 中 制服 ポーズ a 笑顔
; 立ち絵トランジションが実行されます
立ち絵を表示しています。

@しおり 私服
; 立ち絵トランジションが実行されます
立ち絵の服装を変更しています。

@しおり 消
; 立ち絵トランジションが実行されます
立ち絵を消去しています。

@街 夕
; 背景トランジションが実行されます
時間を変更しました。

```

前のページのスクリプトを実行すると、自動トランジションしながら背景や立ち絵が表示されます。背景または時間の変更では envTrans に指定した背景トランジションが実行されます。立ち絵の表示 / 非表示や服装変更では positionTrans、charTrans に指定した立ち絵トランジションが実行されます。タグごとのトランジションの指定が省略できるのでかなり便利です。

```
*start|スタート  
@街 昼 time=2000  
背景を表示しました。  
  
@しおり 前 中 制服 ポーズ a 笑顔 sync  
立ち絵を表示しました。  
  
@しおり 私服 time=2000  
立ち絵の服装を変更しています。  
  
@しおり 消 sync  
立ち絵を消去しました。  
  
@街 夕 nosync  
時間を変更しています。
```

trans 属性で定義済みトランジションを指定した場合と同様に、自動トランジションが行われるタグで一部の属性を指定して変更することができます。@街 昼 time=2000 では、自動トランジションである背景トランジションの time 属性を変更し、2 秒間の非同期クロスフェードトランジションが行われます。@しおり 消 sync では立ち絵トランジションの sync 属性が変更され、0.5 秒の同期クロスフェードトランジションが行われます。

```
*start|スタート  
@街 昼 trans=scroll from=top stay=nostay time=1000  
背景を表示しました。  
  
@しおり 前 中 制服 ポーズ a 笑顔 notrans  
立ち絵をトランジション無しで表示しました。
```

@街 昼 trans=scroll from=top stay=nostay time=1000 のように trans 属性を指定してしまえば自動トランジションは使われません。trans=crossfade の省略である fade 属性でも OK です。自動トランジションと関係なく個別に好きなトランジションが実行できます。

また、@しおり前中制服ポーズa笑顔notransのように「notrans」属性を使用すると、自動トランジションが設定されていてもトランジション無しで表示させることができます。

■フェード時間の定義

```
// ■ fadeValue  
// フェードのデフォルトの時間を指定します。  
fadeValue = 500;  
  
// ■ charFadeValue  
// 立ち絵のフェードのデフォルトの時間を指定します。  
charFadeValue = 500;
```

それぞれミリ秒単位で指定します。「charFadeValue」は立ち絵、「fadeValue」はその以外の画像の場合のデフォルト値を指定します。500ミリ秒から変更する場合は書き換えて「autosetting.bat」を実行してください。

```
*start|スタート  
@街 昼 fade  
背景をフェード表示しました。  
  
@しおり 前 中 制服 ポーズ a 笑顔 fade  
立ち絵をフェード表示しました。
```

fade属性の属性値を省略すると、先ほど指定した時間でのクロスフェードになります。`@街 昼 fade`では`fadeValue`、`@しおり 前 中 制服 ポーズ a 笑顔 fade`では`charFadeValue`に指定した時間になります。

■バックアップ

パソコンのデータはハードディスクに保存されていますが、ハードディスクは消耗品です。ゲーム開発中は必ずバックアップをとりましょう。データが消えてバックアップも無ければ全部最初からやり直しになってしまいます。

バックアップ先としては外付けハードディスクやDVD、USB、オンラインストレージなどがあります。何でもいいので好きなものを使ってください。ここで重要なのはバックアップを自動化することです。万が一に備えて毎日毎日手動で頑張るのはよっぽど真面目な人だけです。データが消えるのは面倒臭がってしばらくさぼっているときです。最初から、とはなりませんが何週間も前からやり直しになります。自動化てしまえばそんな悲しみとも無縁でいられます。

実際のやり方は各自調べて何とかしてください。後でいいや、と思ったらアウトです。やりません。消えてからやり始めることになります。「賢者は歴史に学び、愚者は経験に学ぶ」という言葉があります。私は愚者でした。

自分は使ったことがないのですが、最近の外付けハードディスクには自動的にバックアップしてくれるソフトウェアも付属しているようです。DropBox や SugarSyncなどのオンラインストレージでは、ただ保存しておけば自動的にオンライン上にバックアップを取ってくれます。

■データのやり取り

複数人で協力してゲームを製作していると、ファイルが大量発生して大変なことがあります。直接会って同じ場所で作っていたら別かもしれません、メールやスカイプ越しでファイルをやり取りしていると凄いことになります。何度も送り間違えたかもしれません。そんなことしねーよ、と思うかもしれませんがするのです。

メール越しで毎回ゲームのデータ全部を送るわけにもいかないので、前回送った分から変更されているファイルだけ送るようにしたとします。変更したデータ全てを正確に送り、全てを正確に上書きしてくれればいいのですが、そんなことは無理です。どこかで1度でも間違えばどんどんおかしくなっていきます。2人だけならまだいいですが3人4人5人と人間が集まれば誰かがミスります。

複数人での製作なら最低でも DropBox くらいの機能は導入してください。全員のデータの状態を同じにでき、バックアップも取れてファイルの変更履歴も取れます。お手軽で便利なのですが容量に比して少し値段が高めかなあとはおもいます。50GB で足りるのであれば月10ドルで良い感じです。

■バージョン管理

バージョン管理システムというものがあります。Subversion、Git、Mercurial、Bazaarなど。デジタルノベル製作で実際に使われているのは Subversion になると思います。バージョン管理とは、ファイルの変更の記録を管理することです。

プログラミングをする際には特になのですが、あるファイルを昔の状態に戻したいということがあります。これはバージョン管理システムを使っていれば簡単にできます。吉里吉里のスクリプトを書いていても、書き換えてみたら動かなくなつたけど直し方がわからない、なんてことがあったりします。変更前のファイルに戻てしまえば一発で解決です。

また、同じ「01.ks」を2人で同時に編集していたら困ったことになるのはわかる

とおもいます。この状態を競合といいますが、このような際にもお知らせしてくれたりします。DropBoxなどのオンラインストレージでは気づかずにそのまま保存されてしまうかもしれません。

同人ゲーム製作では、過去のバージョンを残しておく用途にも使えます。夏コミ版、サンクリ版、冬コミ版、と微妙に違ったバージョンをどうやって残しておきますか？バージョン管理のタグによって簡単に残しておけます。

Subversion ですと TortoiseSVN で比較的簡単にバージョン管理システムが使えます。リポジトリを置くためのサーバーが必要ですが、1人であれば DropBox 上に置いてしまっても動きます。チーム製作ならサーバを借りられればベストですが、Subversion のホスティングサービスなんてのもあるようです。いずれにせよ、バージョン管理って何？という人は1度調べてみるといいです。

5.サウンド操作スクリプト

単純に音を鳴らすだけでなく、フェードやボリューム調整など他の機能も紹介します。キャラクターのボイスについてもこの章で扱います。

■同期 / 非同期再生

*start|スタート

@se雨 nosync

効果音を非同期再生しています。

@se雨 stop

効果音を停止しました。

@se雨 sync

効果音を同期再生しました。

@bgm湖畔の村 loop=false sync

BGMを同期再生しました。

音の再生についても、トランジションやアクションと同じく同期 / 非同期があります。同期再生では再生が終了するまでスクリプトの実行が一時停止されます。非同期再生では一時停止しません。

デフォルトでは非同期再生となっていますが、再生を開始するときにsync属性をつけると同期再生できます。ただし、loop属性によってループ再生をする場合は、再生が終わることがないので必ず非同期再生になります。BGMについても`loop=false`であれば同期再生できますが、あまり使うことは無いと思います。

トランジションやアクションの非同期実行と異なり、nowait属性をつけない非同期再生であってもページ送りで再生が停止されるようなことはありません。再生を止めるにはstop属性を使います。

*start|スタート

@bgm湖畔の村 loop=false

BGMを非同期再生しています。

@bgm sync

BGMの終了を待ちました。

`loop=false` の BGM の再生終了は、`@bgm sync` で待つこともできます。

■フェードイン / アウト

*start|スタート

@bgm湖畔の村 time=2000

BGMをフェードイン再生しています。

@se雨 loop time=2000

効果音をフェードイン再生しています。

@bgm stop time=2000

BGMをフェードアウトしています。

@se雨 stop time=2000

効果音をフェードアウトしています。

再生する際に time 属性でミリ秒単位の時間を指定すると、その時間でフェードイン再生されます。stop 属性で停止する際も time 属性をつけるとフェードアウトできます。BGM、効果音で共通です。

*start|スタート

@bgm湖畔の村 time=2000 sync

BGMを同期フェードイン再生しました。

@bgm stop time=2000 sync

BGMを同期フェードアウトしました。

BGM のフェードイン、アウトについては sync 属性をつけると同期動作させることができます。この場合、BGM の再生が終了するまでではなく、フェードが終了するまでスクリプトの動作が一時停止します。

*start|スタート

@se雨 time=2000 sync

効果音を同期再生しました。

@se雨 loop

効果音をループ再生しています。

@se雨 stop time=2000 sync

効果音を同期フェードアウトしました。

効果音については、再生開始時のフェードインで sync 属性をつけると、フェードについての同期ではなく再生についての同期として動作します。 @se雨 time=2000 sync

では、フェードが終了するまでではなく、再生が終了するまでスクリプトの動作が一時停止することになります。stop 属性で停止する際は BGM と同じく、sync 属性をつけるとフェードアウトが終了するまでスクリプトの実行が一時停止します。

■音量指定

BGM の音量は、「全体音量」と「BGM 音量」から決まります。効果音の音量は「全体音量」と「効果音音量」から決まります。全体音量は BGM、効果音、ボイス、ムービーの全ての音量にかかる、文字通りゲーム全体での音量設定です。上部メニューの「音量設定」の「全体音量」からプレイヤーが設定できるようになっています。「BGM 音量」と「効果音音量」では BGM と効果音の音量をそれぞれ調整できます。こちらも「音量設定」メニューから設定できます。

これらはプレイヤー側が設定できる値ですが、さらにスクリプト中で個別に音量調整することもできます。

```
*start|スタート  
@bgm湖畔の村 fade=80  
80%の音量で BGMを再生しています。
```

```
@bgm fade=40  
BGMの音量を 40%にしました。
```

```
@bgm stop  
@se雨 loop  
BGMを停止して効果音を再生しています。
```

```
@se雨 fade=50  
効果音の音量を半分にしました。
```

```
@se雨 fade=100  
効果音の音量を元に戻しました。
```

「fade」属性で再生ごとに個別の音量を設定できます。100 を指定するとそのままの音量、50 を指定すると半分の音量、0 を指定すると無音になります。100 より大きい数字を指定することはできません。再生開始時か再生途中かに関わらず音量を変更できます。

fade 属性で指定する音量は「全体音量」「BGM 音量」「効果音音量」とはまた別のものです。それら 3 つでプレイヤーによって設定された音量から、さらにどれだけ

の音量とするかを指定します。実際にBGMが再生される音量は、「全体音量」「BGM音量」「fadeで指定された値」が掛け算された値となります。3つ全て100の時の音量を $100 \times 100 \times 100 = 1,000,000$ とすると、3つ全て50の時の音量は $50 \times 50 \times 50 = 125,000$ で1/8となります。効果音についてもBGM音量が効果音音量となるだけで全く同じです。

スクリプト中はこんな面倒なことを考える必要はありません。「全体音量」「BGM音量」「効果音音量」はプレイヤーが好きに調整する値なので気にせず、fade属性に指定する値だけを考えればいいです。*fade=50*とすればデフォルトの*fade=100*の時の2分の1の音量となります。*fade=10*なら10分の1です。

```
*start|スタート  
@bgm湖畔の村  
BGMを再生しています。
```

```
@bgm fade=50 time=2000  
BGMの音量を2秒間のフェードで半分にしました。
```

```
@bgm stop  
@se雨 loop  
BGMを停止して効果音を再生しています。
```

```
@se雨 fade=50 time=2000  
効果音の音量を2秒間のフェードで半分にしました。
```

```
@se雨 fade=100 time=2000  
効果音の音量を2秒間のフェードで元に戻しました。
```

「fade」属性を「time」属性と一緒に使うことで、いきなり音量を変更せずにフェードしながら音量を変更できます。

*start|スタート

@bgm湖畔の村 fade=50 time=2000 sync

半分の音量で BGMを再生開始しました。

@bgm fade=100 time=2000 sync

BGMの音量を 2秒間の同期フェードで元に戻しました。

@bgm stop

@se雨 fade=50 time=2000 sync

;効果音再生時の「sync」なので再生終了までの同期動作

BGMを停止して効果音を再生しました。

@se雨 loop

効果音を再生開始しました。

@se雨 fade=50 time=2000 sync

;効果音再生途中の「sync」なのでフェード終了までの同期動作

効果音の音量を 2秒間の同期フェードで半分にしました。

fade 属性と time 属性によるフェードも sync 属性をつけることで同期動作にできます。このときの動作はフェードイン、アウトと同じです。つまり、BGM については sync 属性をつければ音量フェードが終わるまでスクリプトの実行が一時停止します。効果音については、再生開始時に sync 属性をつけると効果音再生終了までの同期動作となります。再生途中でのフェードで sync 属性をつけるとフェード終了までの同期動作となります。

1つ注意として、fade 属性の値は次の再生開始時に fade 属性が省略された場合 100 に戻されます。`@se雨 loop` では、直前に `@se雨 fade=50 time=2000 sync` で fade 属性に 50 が指定されているため半分の音量で再生されるように思うかもしれません。しかし一度再生は終了しているので fade の音量は 100 に戻されて再生します。ここでも 50 で再生したければ `@se雨 loop fade=50` のように再び fade 属性を指定する必要があります。ずっと前に指定した fade 属性の値が残っていたりするとバグのもとになるので、再生開始ごとに 100 に戻されるようになっています。これは BGM についても同じです。

■音量設定について

前提として、BGM や効果音は WAV ファイルの段階でバランスよく音量を調整しておきます。fade 属性はファイルごとの音量のバランスを整えるための機能ではありません。「全体音量」「BGM 音量」「SE 音量」の初期値（ゲームを初めて実行した時に設定されている値）も開発ツールで設定できるので、そちらも良い感じに設定しておきます。せっかくファイル間の音量バランスは良くしても、ゲーム全体で音量が大きすぎれば起動時に爆音が流れてびっくりします。逆に小さすぎて聞こえないというのも問題です。「全体音量」「BGM 音量」「効果音音量」が全て 100 のときにちょうど良くなるようにファイル自体の音量を調整してしまうことが多いです。本当は 3 つ全ての初期値を「50」にして、それで音量がちょうど良くなるように WAV ファイルの音量を調整するのがベターです。こうするとプレイヤーが音量を大きくも小さくもできます。初期設定が 100 では小さくすることしかできないので少し不便かもしれません。

fade 属性の使いどころは、ゲーム中で動的に音量を変更しなければならない場合です。「動的」というのは状況に応じて動作などが変わることです。対義語は「静的」となりますが、こちらはどのような状況でも動作などが変わらないことです。

ゲームについて言えば、プレイヤーの動作で変わっていくゲームの状態は動的な要素となります。変わらない物は静的な要素となります。アクションゲームの敵キャラクタはプレイヤーによって倒されたりされなかつたりする動的な要素です。ゲーム内の地形などは変化しない静的な要素となります。最近は地形も破壊できたりと動的な要素であることもありますがそれはおいておきます。動的な要素の存在は他のメディアと比較してのゲームの特徴です。映画は完全に静的なメディアです。観客が涙を流そうが爆睡しようが映画の内容は寸毫も変わりません。

デジタルノベルは他のジャンルのゲームと比較して動的な要素が少ないため「静的ゲーム」と呼ばれることもあります。アクションゲームなどは「動的ゲーム」です。ただ、デジタルノベルにも動的な要素は存在します。典型的には選択肢によって変化すればそれは動的な要素といえます。

```
*start|スタート  
@se雨 loop  
音量を選択して下さい  
  
@seladd text=50% target=*50パーセント  
@seladd text=100% target=*100パーセント  
@select
```

```
*50パーセント  
@se雨 fade=50  
音量を 50%にしました。
```

```
@next target=*start  
*100パーセント  
@se雨 fade=100  
音量を 100%にしました。
```

```
@next target=*start
```

上のスクリプトではプレイヤーの選択によって効果音の音量を変更しています。このような場合は音量が異なるファイルを2つ準備するよりも fade 属性を使ったほうが便利だと言えます。直接選択肢によらずとも、キャラクタ的好感度やゲームの進行度によって音量が変わる、ということはあり得るかもしれません。

次は選択肢ではなく、プレイヤーの単純なクリックによる動的な動作の例です。

```
*start|スタート  
@bgm湖畔の村  
BGMを再生しています。
```

再生中です。

```
@bgm fade=50  
音量を半分にしました。
```

再生中です。の後にクリックすると次に進み音量が変更されますが、プレイヤーのクリックタイミングによって曲のどの部分から音量を変更するのかが動的に変わってきます。曲の先頭から5秒後からかもしれませんし10秒後からかもしれません。その全ての場合の音楽ファイルを準備することはできないので fade 属性を使って音量を変化させます。

fade 属性は「@se 雨を使う際は常に fade=50 にする」というような使い方をするものではありません。それは wav ファイルの段階で調整してください。fade 属性はゲーム中で変化させる必要がある場合のみに使用します。これは次のパンについても同じです。常に右から左に移動するのであれば、そのような wav ファイルを作れば良いのです。

■パン

*start|スタート

@se雨 loop

効果音を再生しています。

@se雨 pan=100

右から聞こえるように設定しました。

@se雨 pan=-100

左から聞こえるように設定しました。

@se雨 pan=0

中央に戻しました。

効果音について、「pan」属性を使うとパン(音の聞こえてくる方向)を指定できます。プラスの値を指定すると右、マイナスの値を指定すると左から聞こえてくるようになります。デフォルトでは0で、-100～100の値が指定できます。pan属性は今のところ効果音のみでBGMでは使用できません。もしかしたら今後使えるようになるかもしれません。

*start|スタート

@se雨 loop

効果音を再生しています。

@se雨 pan=100 time=2000

音を右に移動させています。

@se雨 pan=-100 time=2000 sync

音を左に移動させました。

@se雨 pan=0 time=2000 sync

中央に戻しました。

time属性を使うこともできます。指定された方向まで指定された時間をかけて聞こえてくる方向が移動します。sync属性で同期動作させることもできます。注意点もfade属性と同じです。再生開始時のsync属性は再生終了までの同期動作となります。pan属性は次の再生開始時に100ではなく0に戻されます。

ここまで紹介してきた音のsyncによる同期動作では「canskip」属性が使用できます。falseを指定するとクリックで同期動作をスキップすることができなくなります。

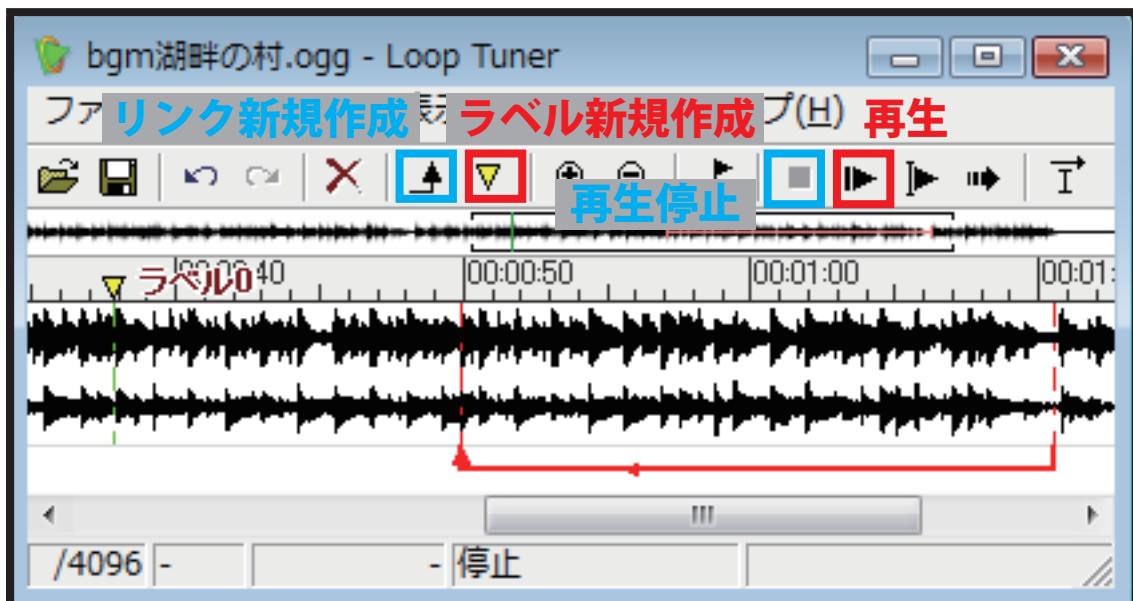
デフォルトでは true となっているのでスキップできます。

■ループチューナ

吉里吉里には「ループチューナ」という開発ツールが付属しています。この使い方を少しだけ解説します。先頭から末尾のループ以外にも途中でループさせたり、音楽ファイルにラベルという印を付けることができたりします。

tools フォルダの「krkrlt.exe」がループチューナです。起動してみてください。「ファイル」メニューの「開く」から設定するファイルを選択します。ここでは例としてプロジェクトフォルダの「bgm」フォルダに入っているはずの「bgm 湖畔の村.ogg」を開きます。

メニューの右の方の黒い左向きの三角のボタンで現在のファイルを再生できます。黒い四角のボタンで再生停止します。



メニュー真ん中あたりの黄色い下向き三角のボタンでラベルが新規作成できます。クリックすると下の波形の上に黄色い三角が出てきたと思います。それがラベルです。何個でも作れます。作ったラベルは、波形の上の黄色い三角をドラッグすると位置を変更できます。ダブルクリックで名前を変えることもできます。

リンクを作るには、ラベル新規作成ボタンの左隣のボタンをおします。一番下に赤い矢印が出てきます。それがリンクです。リンクしたところでループするようになります。

作ったラベルとリンクは「ファイル」メニューの「保存」で保存できます。保存す

ると「bgm 湖畔の村 .ogg」と同じフォルダに「bgm 湖畔の村 .ogg.sli」というファイルができますが、それがループチューナのデータです。

詳細なループチューナの使い方は吉里吉里リファレンスの説明 (<http://devdoc.kikyou.info/tvp/docs/kr2doc/contents/LoopTuner.html>) を参照してください。私も勘と慣れでテキトーに使っているので説明できません。

音を作っている人などがいればその人に任せてしまうのがいいです。おそらく使いこなしてくれるでしょう。

ひとまずサンプルとして使いたいので「湖畔の村 .ogg」はリンクを 1 つと「ラベル 0」という名前のラベルを 1 つ作成したら保存してください。

音のプロフェッショナルな方はもっと良い専用ツールを使って「サンプル数」なるものでループ位置を指定してきます。この場合は設定ファイルを直接いじってしまいます。「bgm 湖畔の村 .ogg」と同じフォルダにある「bgm 湖畔の村 .ogg.sli」をテキストエディタで開いてください。

```
Link { From=2504685; To=1525930; Smooth=False; Condition=no;  
RefValue=0; CondVar=0; }
```

ここでは長すぎて途中で改行されますが、上のような行があると思います。「From=」の後の数字がループ元のサンプル数、「To=」の後の数字がループ先のサンプル数になります。ここでは *From=3112941;* と *To=1333418;* に書き換えて保存してください。きれいにループするようになります。ちゃんとループしているか、もう 1 度ループチューナで「bgm 湖畔の森 .ogg」を開いて再生してみると確認できます。

ループチューナの設定が保存された拡張子が sli のファイルは、元の音楽ファイルと同じフォルダに置いておけば、再生時に吉里吉里が勝手に読み込んでくれます。

```
*start|スタート  
@bgm湖畔の村  
BGMを再生しています。
```

ただ BGM を再生しているだけですが、いつまでも途切れずにループチューナで設定されたリンクの位置でループされているのがわかると思います。

```
*start|スタート  
@bgm湖畔の村 start=ラベル 0  
SEを再生しています。
```

「start」属性にループチューナで設定したラベルの名前を指定すると、そのラベルの位置から再生できます。効果音でもラベルが設定してあれば start 属性が使えます。

■ボイス再生

同人ゲームでもボイスの入っているゲームは珍しくなくなっています。KAGEX では標準でボイス再生のための機能がついています。

サンプル素材フォルダの「voice」フォルダに入っている「cv しおり _0001.ogg」などのファイルを、全てプロジェクトフォルダの「voice」フォルダにコピーしてください。ボイスファイルのファイル名は、「cv[キャラ名]_[4桁の番号]」という形になります。

```
*start|スタート  
@しおり voice=1  
【しおり】「こんにちは」
```

【しおり】「ボイス再生のテストです。」

名前をつけなければボイスは再生されません。

【しおり】「ボイスの番号は自動的に1ずつ上がっていきます。」

ボイスを再生するには char タグを使います。char タグは立ち絵だけでなく、キャラクタの操作全般に使用します。ここではキャラ名をタグ名にする省略記法を使っています。

「voice」属性にボイスの番号を指定すると、次に【しおり】で名前を表示するときにその番号のファイルが再生されます。【しおり】「こんにちは」で「cv しおり _0001」が再生されることになります。そしてその次の名前表示では、その前の番号に1を足した番号のボイスが自動的に再生されます。【しおり】「ボイス再生のテストです。」では「cv しおり _0002」、【しおり】「ボイスの番号は自動的に1ずつ上がっていきます。」では「cv しおり _0003」が自動的に再生されます。

```
*start|スタート  
@しおり voice=1  
;cvしおり _0001 が再生されます  
【しおり】「こんにちは」  
  
@かえで voice=1
```

```
;cvかえで _0001 が再生されます
```

【かえで】「こんにちは」

```
;cvしおり _0002 が再生されます
```

【しおり】「ボイス再生のテストです。」

```
;cvかえで _0002 が再生されます
```

【かえで】「ボイスの番号はキャラクタごとに管理されています。」

```
;cvかえで _0003 が再生されます
```

【かえで】「他のキャラクタの番号には影響されません。」

ボイスの番号はキャラクタごとにそれぞれ管理されています。コメントで書いてある通りに再生されます。キャラクタが3人以上いても同じです。

```
*start|スタート
```

```
@しおり voice=1
```

```
;cvしおり _0001 が再生されます。
```

【しおり / ? ? ?】「こんにちは」

テキスト表示でやりましたが、半角スラッシュの後に実際に表示する名前を指定できます。半角スラッシュの前の名前はボイス再生の際にどのキャラクターを判定するのに使われています。

```
*start|スタート
```

```
@しおり voice=1
```

```
;cvしおり _0001 が再生されます
```

【しおり】「こんにちは」

```
@しおり voice=5
```

```
;cvしおり _0005 が再生されます
```

【しおり】「番号は指定すればいつでも変えられます。」

```
@しおり voice=ボイス 1
```

```
;ボイス 1 が再生されます
```

【しおり】「ファイル名で指定することもできます」

```
;cvしおり _0006 が再生されます
```

【しおり】「ファイル名が指定されたセリフでは番号が足されません。」

ボイスファイルの番号が連続していない部分は voice 属性にその都度番号を指定します。また、ボイスに追加があった場合など、ファイル名が特殊な場合はそのファイル名を指定して再生することもできます。

```
*start|スタート
@しおり voice=1
;cvしおり _0001 が再生されます
【しおり】「こんにちは」

@しおり voice=ignore
【しおり】「ここではボイスが再生されません。」

;cvしおり _0002 が再生されます
【しおり】「ボイス再生のテストです。」
```

voice 属性に「ignore」を指定すると、次のそのキャラの名前表示でボイスが再生されません。その次からは通常通り再生されます。

```
*start|スタート
@しおり voice=1
;cvしおり _0001 が再生されます
【しおり】「こんにちは」

@しおり voice=clear
【しおり】「これ以降、しおりのセリフでボイスが再生されません」

【しおり】「ここでも再生されません。」

【しおり】「ボイス無しです。」

@しおり voice=2
;cvしおり _0002 が再生されます
【しおり】「ボイス再生のテストです。」
```

voice 属性に「clear」を指定すると、次以降のそのキャラの名前表示でボイスが再生されなくなります。また voice 属性で番号を指定すればボイス再生を再開できます。

```
*start|スタート
@しおり playvoice=2 nosync
;cvしおり _0002 が再生されます
```

名前表示に関係なく再生できます。

```
@しおり playvoice=ボイス1  
;ボイス1が再生されます  
ファイル名で指定することもできます。
```

```
@しおり playvoice=3 waitvoice  
;cvしおり_0003が同期再生されます  
ボイスを同期再生しました。
```

voice 属性は次の名前表示で再生するボイスを指定する属性でしたが、「playvoice」属性を使うとその場ですぐにボイスファイルを再生できます。voice 属性と同じく番号かファイル名で指定します。playvoice 属性では sync 属性で同期再生することはできません。代わりに「waitvoice」属性が使えます。

音声を再生するだけなら、効果音を再生するように `@se play=cvしおり_0002` でもできてしまいますが、ボイス扱いのものは voice 属性や playvoice 属性で再生するようにしてください。効果音として流すと効果音音量が適用されるため、ボイス音量が適用されなくなります。

ボイスの音量は「全体音量」「ボイス全体音量」「ボイス個別音量」で決まります。全体音量は BGM や効果音と共通です。「ボイス全体音量」は「ボイス設定」メニューから設定できます。「ボイス個別音量」は、キャラごとの音量設定です。デフォルトではメニューに表示されていませんが、ツールでキャラ名を設定すると「ボイス設定」メニューに表示させることもできます。

```
*start|スタート  
@しおり playvoice=ボイス1  
ボイス再生中です。  
  
@しおり stopvoice  
ボイスを強制的に停止しました。
```

「stopvoice」属性を使うと、そのキャラクタのボイスを強制的に停止することもできます。上のスクリプトで、ボイスが終わる前にボイス再生中です。をクリックで飛ばすと、ボイスが途中でも停止します。

■遅延実行

一旦ボイスからは離れますが、「遅延実行」はボイスと共によく使われる所以ここで解説します。遅延実行ではタグを実行するタイミングを遅延させることができます。

```
*start|スタート  
@しおり 前 中 制服 ポーズ b 無表情  
立ち絵を表示しました。
```

```
@しおり 左  
@しおり 笑顔 delayrun=2000  
立ち絵の表情を変更します。
```

「stage」「char」「event」「layer」「env」「bgm」「se」タグでは「delayrun」属性が使えます。すべてのタグは上から順番に実行されますが、「delayrun」属性にミリ秒単位の時間を指定することで、その効果が現れるタイミングをずらすことができます。`@しおり 左`ではすぐに立ち絵が左に移動しますが、`@しおり 笑顔 delayrun=2000`では2秒後に表情が変わることになります。delayrun 属性がついたタグは書かれている場所では一見無視され、指定された時間が経過するまでその効果が現れません。

```
*start|スタート  
@layer name=星 1 file=星 show  
星を表示しました。  
  
@星 1 xpos=300 time=3000  
@星 1 opacity=0 time=1000 delayrun=2000  
星は2秒後から消え始めます。
```

表情変更に限らず遅延実行できます。上のスクリプトでは`@星 1 opacity=0 time=1000`というタグが delayrun 属性によって2秒後に実行されます。結果として、星は2秒後から1秒かけて透明になっていくことになります。

```
*start|スタート  
@しおり 前 中 制服 ポーズ b 無表情  
立ち絵を表示しました。  
  
@しおり 左  
@しおり 右 delayrun=10000 sync  
立ち絵を10秒後に右に移動します。
```

遅延実行はページ送りでキャンセルされます。

nowait 属性をつけない非同期アクションや非同期トランジションはページ送りでキャンセルされましたが、遅延実行も同じくページ送りでキャンセルされます。すぐに遅延実行されるタグの実行終了後の状態になります。遅延実行で nowait 属性は使えないで必ずキャンセルされることになります。また、遅延実行を sync 属性などで同期動作にすることもできません。必ず非同期動作になります。

```
*start|スタート  
@しおり 前 中 制服 ポーズ b 無表情  
立ち絵を表示しました。  
  
@しおり voice=10  
@しおり ポーズ a 笑顔 delayrun=ラベル 0  
【しおり】「ボイスに合わせてタグを実行することができます。」
```

delayrun 属性にボイスファイルのラベル名を指定すると、その再生に合わせてタグを実行できます。「cv しおり _0010.ogg」をループチューナで開くとわかりますが、「タグを実行することができます。」というセリフの「タグ」の直前に「ラベル 0」という名前のラベルが挟まれています。ボイスがその部分まで再生されたときに@しおり ポーズ a 笑顔が実行されることとなります。ラベルを複数置けばセリフ中に何個でも遅延実行させることができます。ボイスが指定されたラベル位置まで再生される前にページ送りが発生した場合は、時間を指定した場合と同じように遅延実行はキャンセルされます。

■表情指定のスクリプトとか

遅延実行で最もよく使うのは最後に紹介したボイスの再生に合わせて表情を変更したりアクションを実行したり、という場合になると思います。便利ですが遅延実行を使用するとスクリプトの作業量が跳ね上がる所以覚悟したほうがいいです。途中で力尽きたと前半は動きまくるのに後半は何だかしょぼい、というゲームになりかねません。

「デバッグ」メニューの「ループチューナ」をクリックすると、最後に再生したボイスファイルをループチューナで開くことができます。「Shift+C」のショートカットキーでも開けます。これを活用するとボイスファイルにラベルをつけるのが楽になります。大量にあるボイスファイルからお目当てのファイルを探し出すより、それが使われる場面までゲームを進めて Shift+C の方が早いです。

実際のスクリプト作業としては立ち絵の表情変更が大部分を占めことが多いです。最初にテキストとボイスの番号の指定を済ませたスクリプトを用意して、それを実行しながらボイスのラベル付けやスクリプトの編集をしていきます。この際、編集

する部分の直前でセーブしておくと、ゲームを再起動する代わりにロードするだけで動作が確認できるので便利です。ロードの際にロード先のスクリプトを読み込みますが、これを利用して編集済みのスクリプトを読みこませることができます。編集前と編集後でラベルの位置が違っていたりするとおかしくなる場合もあるので、この時は一度ゲームを再起動してセーブしなおしになります。

「デバッグ」メニューの「立ち絵参照ウィンドウ」を使うと立ち絵の表情の組み合わせを見ながら作業できるのでもうちょっと便利になります。「立ち絵変更」という小さいウィンドウの方を閉じてしまった場合は「ファイル」メニューの「サブウィンドウ表示」から再度開けます。使い方はいじっていればなんとなくわかってくると思います。

ゲーム本体のウィンドウ、KAGEX ログ、立ち絵参照ウィンドウ、立ち絵変更ウィンドウ、ループチューナ、テキストエディタと開いていくと、ウィンドウがどんどん増えてきます。これに加えてコンソールや画像編集ソフトやエクセルのファイル名対応表なども開いたりします。作業中は Twitter や Skype は閉じましょう。まあ、まともにやろうとするとモニタ 1 枚では足りないです。デュアルモニタおすすめです。執筆時点では自分も 2 枚ですが、お金ができたら 3 枚以上にする予定です。印刷できるものは紙に印刷してモニタの横に貼っておく、というのもいいと思います。

6.メッセージ操作スクリプト

テキスト表示に関するスクリプトを紹介します。テキストの位置を変更したり、テキスト枠として画像を表示させたりします。

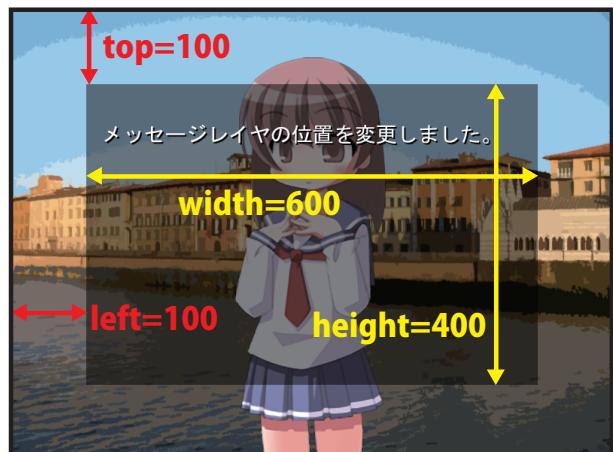
■テキストの表示位置

```
*start|スタート  
@街 昼  
@しおり 前 中 制服 ポーズ b 無表情  
背景と立ち絵を表示しました。
```

```
@position layer=message0 page=fore left=100 top=100 width=600 height=400  
color=&RGB(0,0,0) opacity=127  
;長すぎて途中で改行されてますが1行に書いてください。  
メッセージレイヤの位置を変更しました。
```

メッセージレイヤの位置やサイズは「position」タグで変更することができます。「layer」属性と「page」属性で、どのメッセージレイヤの設定を変更するのかを指定します。layer 属性にはメッセージレイヤの番号を指定します。メッセージレイヤは1つだけでなく複数使用する事ができるのですが、これまで使っていたのは0番なので *layer=message0* とします。番号の前には必ず「message」と付きます。page 属性には表ページの場合「fore」、裏ページの場合は「back」を指定します。メッセージレイヤには「表ページ」と「裏ページ」というものがあります。これについては後述します。とりあえず *page=fore* としておきます。

「left」属性と「top」属性に、それぞれ横、縦方向のメッセージレイヤ位置を指定します。ウィンドウ左上端からのピクセル数を指定する形になります。「width」属性と「height」属性にはメッセージレイヤのサイズを指定します。それぞれ幅、高さをピクセル単位で指定します。



「color」属性には *color=&RGB(赤,緑,青)* の形でメッセージレイヤの色を指定します。赤、緑、青のそれぞれに0～255で色の強さを指定します。ここでは *color=&RGB(0,0,0)* なので真っ黒になります。「opacity」属性にはレイヤの不透明度を設定します。立ち絵などで指定できる *opacity* 属性と同じく0～255を指定します。ここでは *opacity=127* で半透明にしています。

```
*start|スタート
```

```
@街 昼
```

```
@しおり 前 中 制服 ポーズ b 無表情
```

```
@position layer=message0 page=fore left=100 top=100 width=600 height=400  
color=&RGB(0,0,0) opacity=128 marginl=100 margint=100 marginr=100  
marginb=100
```

メッセージレイヤの位置を変更しました。

「marginl」「margint」「marginr」「marginb」属性で、メッセージレイヤの中のテキスト表示位置を指定します。それぞれ左、上、右、下の余白部分をピクセル単位で指定することで、テキストの位置を決定します。テキストはその余白の内側に表示されることになります。

上のスクリプトの場合、右の画像の赤い枠の内側にテキストが表示されることになります。テキストの右側に1文字分の余裕がありますが、これは禁則処理のための余裕になっています。



```
*start|スタート
```

```
@白
```

```
@position layer=message0 page=fore left=100 top=100 width=600 height=400  
color=&RGB(0,0,0) opacity=128 marginl=100 margint=100 marginr=100  
marginb=100 nameleft=50 nametop=50 namewidth=200 nameheight=50
```

【しおり】「メッセージレイヤの位置を変更しました。」

「nameleft」属性と「nametop」属性で、それぞれ横、縦方向の名前欄の位置を指定します。メッセージレイヤ左上端からのピクセル数で指定します。「namewidth」属性と「nameheight」属性には名前欄のサイズを指定します。それぞれ幅、高さをピクセル単位で指定します。

細かくて見づらいですが右の画像の赤い枠の内側に名前が表示されることになります。



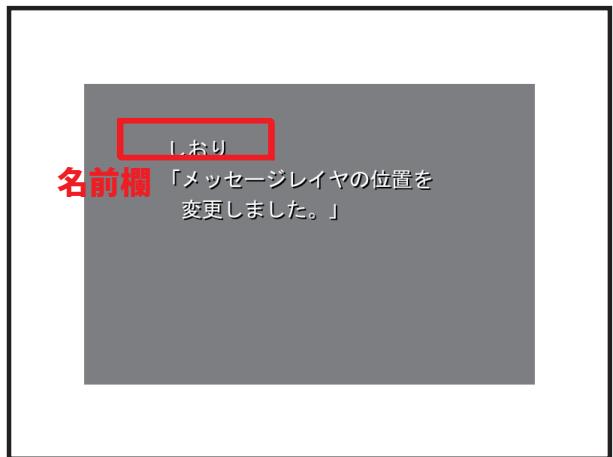
```
*start|スタート
```

```
@白
```

```
@position layer=message0 page=fore left=100 top=100 width=600 height=400  
color=&RGB(0,0,0) opacity=128 marginl=100 margint=100 marginr=100  
marginb=100 nameleft=50 nametop=50 namewidth=200 nameheight=50  
namealign=0 namevalign=1
```

【しおり】「メッセージレイヤの位置を変更しました。」

「namealign」属性と「namevalign」属性で、名前欄の中で名前をどこに表示するかを指定します。namealign 属性は横方向の位置で、「-1」で左寄せ、「0」で中央寄せ、「1」で右寄せになります。namevalign 属性は縦方向の位置で、「-1」で上寄せ、「0」で中央寄せ、「1」で下寄せとなります。上のスクリプトでは *namealign=0 namevalign=-1* で横方向は中央、縦方向は下に寄せる設定にしてあります。



■色の指定について

パソコンのモニタでは赤、緑、青の3色の光を組み合わせて全ての色を表現する「加法混色」(RGB カラー)という方法が採られています。赤(Red)、緑(Green)、青(Blue)がそれぞれ0～255の256段階で、最大 $256 \times 256 \times 256 = 16,777,216$ 色表示できます。「約1700万色」というやつです。

それぞれの数字が大きいほどその色の光が強くなります。(赤, 緑, 青)がそれぞれ(255,0,0)なら赤、(0,255,0)なら青、(0,0,255)なら緑となります。(255,255,255)で全ての光の強さを最大にすると白となります。詳しくは加法混色で調べてください。

メッセージレイヤについては、色で指定せず、代わりにテキスト枠の画像を指定してしまうことが多いので color 属性はそれほど使いません。

■テキスト枠

position タグの color 属性では単色の四角いメッセージレイヤしか表示できませんでしたが、画像を使用することで好きな形と色をメッセージレイヤに表示することができます。

サンプルの uiimage フォルダの「message.png」を表示させてみます。プロジェクトフォルダの「uiimage」フォルダにコピーしておいてください。

```
*start|スタート
```

```
@街 昼
```

```
@position layer=message0 page=fore frame=message opacity=255 left=0  
top=400 marginl=60 margint=50 marginr=60 marginb=40 nameleft=60  
nametop=0 namewidth=156 nameheight=37 namealign=0 namevalign=0
```

```
【しおり】「メッセージレイヤの画像を変更しました。」
```

画像を使用する場合は、color属性の代わりに「frame」属性に画像のファイル名を指定します。message.pngは元から半透明の画像なのでopacity属性は255で不透明にしてあります。

left、top画像でレイヤの位置は指定しますが、width、height属性でのサイズ指定は必要ありません。メッセージレイヤはframe属性に指定した画像と同じサイズになります。テキスト位置や名前欄の位置についても画像を使用しない場合と同じように指定します。上のスクリプトでは右の画像のように指定しています。



■メッセージレイヤの表示 / 非表示

```
*start|スタート
```

```
@街 昼
```

```
背景を表示しました。
```

```
@msgoff
```

```
@msgon
```

```
メッセージレイヤを一時消去しました。
```

```
@msgoff fade=1000
```

```
@msgon trans=scroll time=1000 from=top stay=nostay
```

```
メッセージレイヤを一時消去しました。
```

「msgoff」タグを使用すると、メッセージレイヤを非表示にできます。非表示にしたメッセージレイヤは「msgon」タグで再び表示できます。

msgoff、msgonタグではtrans属性やfade属性でトランジションを指定することもできます。このトランジションは必ず同期動作になり、nosync属性を使うことはできません。`@msgoff`のようにトランジションの指定を省略した場合は、自動トラン

ジションのところで紹介した、fadeValue に指定した時間でのクロスフェードトランジションとなります。

```
*start|スタート  
@街 昼  
背景を表示しました。  
  
@msgoff  
メッセージレイヤを一時消去しました。
```

メッセージレイヤを一時消去しました。では、メッセージレイヤが消えた状態でテキストを表示させようとしています。このような場合は、テキストを表示する直前に自動的に @msgon が実行されてメッセージレイヤが表示されるようになっています。

最初のテキストが表示される前はメッセージレイヤが非表示になっています。なので、背景を表示しました。の直前でも @msgon が自動的に実行されトランジションでメッセージレイヤが表示されるような動作になっています。

```
*start|スタート  
@街 昼  
背景を表示しました。  
  
@position layer=message0 page=fore visible=false  
トランジションなしでメッセージレイヤを一時消去しました。
```

@msgoff や @msgon では、自動的にクロスフェードトランジションが行われます。トランジション無しでメッセージレイヤを消去したい場合は position タグの「visible」属性を使用します。true を指定すれば表示、false を指定すれば非表示となります。

```
*start|スタート  
@街 昼  
背景を表示しました。  
  
@夕 trans=crossfade time=1000 msgoff  
時間を変更しました。  
  
@しおり 前 中 制服 ポーズ b 無表情 trans=crossafade time=500 msgoff  
【しおり】「立ち絵を表示しました。」
```

トランジションの所では紹介しませんでしたが、trans 属性によるトランジションでは「msgoff」という属性と一緒に使うことができます。この属性に true を指

定するとトランジションの直前でメッセージレイヤが自動的に消去されます。@*夕 trans=crossfade time=1000 msgoff* は、@*msgoff* と @*夕 trans=crossfade time=1000* を 2 行にわけて書いても同じ動作になります。@*しおり 前 中 制服 ポーズ b 無表情 trans=crossfade time=500 msgoff* も同じく 2 行にわけて書くことができます。

実際にスクリプト中で *msgoff* 属性を使うというよりは、自動トランジションを定義する際に *msgoff* 属性を使用して、背景を変更する際はメッセージレイヤを常に消去する、というような演出をする場合に使うと便利です。

■選択肢

3 章ではかなり簡単にしか説明していなかったので、ここで選択肢について詳しく解説します。

```
*start|スタート  
選択して下さい。  
@seladd text=選択肢 1 target=*選択肢 1  
@seladd text=選択肢 2 target=*選択肢 2  
@select
```

```
*選択肢 1  
選択肢 1 が選択されました。  
@s  
*選択肢 2  
選択肢 2 が選択されました。
```

3 章の復習になりますが、*seladd* タグで選択肢を登録し、*select* タグで選択肢を表示することができます。

選択肢には直接関係ないのですが、「*s*」タグでスクリプトの実行を停止できます。選択肢 1 が選択された際に停止しなければ選択肢 2 が選択されました。の部分まで実行されてしまうので *@s* で停止させています。

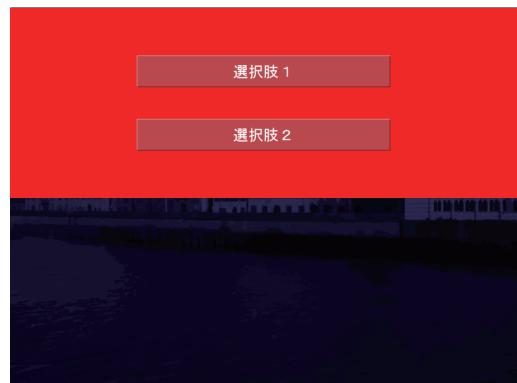
```
*start|スタート  
@selopt left=0 top=0 width=800 height=300  
@seladd text=選択肢 1 target=*選択肢 1  
@seladd text=選択肢 2 target=*選択肢 2  
@select
```

行数の節約のために *選択肢 1 と *選択肢 2 のジャンプ先のスクリプトは省略しています。動作を確認する場合は先程の *選択肢 1 から選択肢 2 が選択されました。までを付け足し

てください。ここから選択肢の説明をしている間は共通です。

選択肢の動作は「selopt」タグを使うことで設定できます。left、top、width、height 属性で「選択肢領域」を指定できます。left 属性と top 属性でそれぞれ横、縦方向の位置、top 属性と width 属性でそれぞれ幅と高さを指定します。

選択肢領域とは選択肢を表示する範囲のことです。その範囲に選択肢が等間隔で並べられます。右の画像ではわかりやすいように領域に赤色を塗っていますが、実際には透明で見えません。デフォルトの選択肢領域は画面全体となっています。



選択肢に画像を使うことができます。サンプルの uiimage フォルダの「選択肢_normal.png」「選択肢_over.png」「選択肢_on.png」をプロジェクトフォルダの「uiimage」フォルダにコピーしてください。

```
*start|スタート
@selopt normal=選択肢_normal over=選択肢_over on=選択肢_on
@seladd text=選択肢 1 target=*選択肢 1
@seladd text=選択肢 2 target=*選択肢 2
@select
```

selopt タグの「normal」「over」「on」属性に選択肢に使用する画像のファイル名を指定します。それ respective 通常時、マウスがボタンに乗った時、ボタンがクリックされた時の画像を指定します。ここで指定した画像の上に seladd タグの text 属性に指定したテキストが描画されて表示されます。

```
*start|スタート
@selopt fadetime=1000
@seladd text=選択肢 1 target=*選択肢 1
@seladd text=選択肢 2 target=*選択肢 2
@select
```

sekiot タグの「fadetime」属性で、選択肢を表示する際のフェードの時間を指定します。デフォルトでは 0 でフェードなしで表示されます。選択肢を表示する際のフェードイン、選択後に消去する際のフェードアウトの時間を共通で指定します。

*start|スタート

@selopt normal=選択肢 _normal over=選択肢 _over on=選択肢 _on
選択肢の設定をしました。

@seladd text=選択肢 1 target=*選択肢 1

@select

*選択肢 1

選択肢 1 が選択されました。

@seladd text=選択肢 2 target=*選択肢 2

@select

*選択肢 2

選択肢 2 が選択されました。

selopt タグで指定した設定は、それが変更されるまで変わりません。最初に 1 度だけ指定しておけばそれ以降の選択肢で使用されます。*@select* ごとに *@selopt* タグを使う必要はありません。上のスクリプトでは、最初の *@selopt* による画像が両方の選択肢で使用されます。

■後書き

凄い中途半端なところでぶつた切りますが、ここでタイムアウトです。現在、頒布前日の12月30日22時を回ったところです。

最初は初心者向けに書いていたのですが途中から何が何やら、どんどん残念な方向に。そこらへんの言い訳はブログやTwitterを見ると残ってるかもしれません。ここでも申し開きしたいところですが我慢します。次の夏コミにはちゃんとしたものが出したいものです。

まだ準備できていないのですが、冬コミ後にはこの本で使うSDKを公開します。遅くとも1週間以内になんとかします。そちらの方に説明書と言う名の補足が付いている可能性があるのでチェックするようにお願いします。本文のどこかにも書いてありますが、本書のサポートウェブページは、<http://biscrat.com/book/kagex/support/>となります。

次の章は大幅に方針転換しまして、KAGを使ってる方向けに、KAGEX使う上で参考になりそうな話を詰め込んでおきます。少しでも役立つように思いついたところを片っ端から書いていきます。完全に今から考えるのでひどい事になるかもしれませんがご了承ください。

7.おまけ

■オブジェクトについて

本編の最初の方で stage、char、layer、event というタグを直接使ってましたが、実際には使わないでください。ほぼ間違いなく省略記法で使います。初心者向けにタグの説明したいということで丁寧にやってみましたが今振り返ると失敗でした。

それらは簡単にするために「画像を表示するためのタグ」として紹介しました。実際は stage は舞台オブジェクト、char はキャラクタオブジェクト、layer は環境レイヤオブジェクト、event はイベントオブジェクトを操作するためのタグです。それに加えて env が環境オブジェクトを操作するタグ、bgm が BGM オブジェクトを操作するタグ、se が効果音オブジェクトを操作するタグです。

KAGEX ではデジタルノベルで使用される概念を「オブジェクト」としてまとめています。KAG だとシステム的にもっと低レベルで、image タグでレイヤを直接扱つたりしました。KAGEX ではそんなことはしません。

特にキャラクタが例として分かりやすいですが、char タグで操作できるキャラクタオブジェクトには、キャラクタとして必要な操作がまとまっています。立ち絵表示、ボイス再生、表情画像表示などがまとめて char タグ 1 つでつかえます。内側では何枚ものレイヤと再生バッファを扱っていて、KAG のプリミティブな操作と比べるとかなり高レベルな実装になっています。

ところで低レベル、高レベルというのはプログラム的な意味での言葉なのであしからず。低レベルな処理というのは、色々できますが色々やらないと駄目という意味で、高レベルな処理というのは用途は特定されてるけども便利な機能が簡単に使えるという感じの意味です。

■補足

立ち絵については結局表情結合済みの立ち絵しか扱えていません。少し規模が大きくなると、表情部分は分離することが多いと思います。それについての説明は開発ツールに付属するはずなのでそちらを参照してください。

本編でアクション定義について説明できなかったのですが、今回しっかり調べてみたところ思った以上に複雑で説明端折ってしまいました。こちらも開発ツールの方に説明つくと思います。

■シナリオスクリプト / システムスクリプト

本編の説明は、ほぼシナリオスクリプト部分の説明です。KAGEX 使う上ではゲーム本編中の「シナリオスクリプト」と、タイトル画面やセーブ / ロード画面、オプション画面などの「システムスクリプト」はわけて考えたほうがわかりやすいです。システムスクリプトについては、この後の章で扱う予定でしたがカットです。

シナリオスクリプトは、実は本編で説明されている知識だけで十分だったりします。シナリオ中ではテキスト表示と立ち絵の服装や表情変更、加えてたまに効果音やBGM を鳴らすくらいしかしません。それだけ知つれば作れます。

■マクロ

本編でマクロの説明をしていないんですが、KAGEX の素の部分だけで既に必要なだけまとまっています。KAGEX で何本かゲーム作つてますがマクロなんてほとんど使いません。

マクロ使うとしたらフィニッシュ時のフラッシュのような定型のエフェクトつけるくらいですね。

@endtrans についてはマクロにしてしまうと便利です。立ち絵切り替えや背景切り替えでは自動トランジションが登録できるのでいいのですが、@endtrans では自動トランジションが定義できません。

```
@macro name=et  
@endtrans 背景トランジション*  
@endmacro
```

これだけでもいいです。@endtrans のかわりに @et を使えば「背景トランジション」を自動トランジションのように使えます。

マクロの定義は「sysinit」フォルダの「macro.ks」でするようにしてください。少し「first.ks」を覗いてもらえば分かりますが、@call storage=macro.ks となっています。KAG でもマクロ定義スクリプトを複数回読み込んでいる初心者の方がいるんですが、macro.ks は起動直後に1回しか読み込まれない専用スクリプトファイルとして最初から準備してしまっています。

■envinit.tjs

シナリオスクリプトは非常に簡単なのですが、その前段階としてのシステムスクリプトが必要になります。KAG から移行する上でこれが壁になります。本来は「envinit.tjs」というファイルに舞台オブジェクトやキャラクタオブジェクトの内容について、TJS の辞書配列で定義していきます。これが難解なため KAGEX 難しいというイメー

ジがついてしまうのではないかと思う。実際はそこまででもないんですが、リファレンスが本当に皆無なために最初は引っ掛かります。

本書では envinit.tjs の代わりに envinit.otjs を書いています。envinit.tjs を直接書くよりは、config.tjs を書くのに似ていて簡単だとおもいます。それでも面倒なのでツールでできるようにする予定です。SDKがリリースされたら説明書を参照してください。

本書ではもう 1 つの簡略化として、背景や立ち絵などの画像の名前形式を固定化してしまっています。本来はこの名前のルールからして定義しないと駄目なのが初心者お断わりに拍車をかけています。こんなものは固定でいいはずです。SDK を使う場合は本編に書いてあるルールに従って頂ければ OK です。autosetting.bat というのを何度も本編で使いましたが、それでファイルを検索して envinit.tjs に変換しています。

■選択肢

選択肢で seladd、select、selopt を紹介しました。似たような選択肢しか使わないゲームならこれで十分です。気に入らなければ KAG と同じく button タグや link タグが KAG と同じように使えるので活用してください。button タグについては graphic 属性で通常時、マウスオーバー時、マウスクリック時が並んだお馴染みの画像も指定できますが、選択肢と同じく normal、over、on 属性でバラバラにした画像も読み込めるので便利です。

■TJS 的な話

KAG では image とか trans とか playbgm とか、とにかく色々なタグを使っているとおもいます。KAGEEX では 9 割方 char、stage、event、layer、env、bgm、se の 7 つのタグで何とかなります。タグが少ない代わりに属性の種類が多いです。このオブジェクトを操作する属性のことをコマンドなんて言ったりします。名前はどうでもいいですが、system フォルダを commands で grep するとコマンドの動作が追えます。

KAG だと tagHandlers を見ておけばそれぞれのタグがどんな動きをしているかが分かったのですが、KAGEEX ではオブジェクトごとに属性がまとまっています。例えばキャラクタオブジェクトについては「KAGEnvCharacter.tjs」の `var charCommands = %/` 以下に char タグで使える属性の動作が書いてあります。これだけでも知っていると KAGEEX の TJS を読みたい場合は楽になるかもしれません。

ついでに「KAGEnvImage.tjs」の `var commands = %/` 以下は画像を扱えるオブジェクト (stage、char、event、layer) で共通に使える属性の動作となっています。

■変数

本編で一切触れていないのですがゲーム変数 f、システム変数 sf、一時変数 tf は KAG と同じく好きに使ってもらって大丈夫です。if タグでの条件分岐なども必要であればご自由にどうぞ。最近そういったフラグ管理が必要なゲームはあまり作ったことがないんですが。ほぼ一本道か選択肢何個か選んで～程度のゲームばっかり作ってます。

■システムスクリプト

SDK ではシステム部分も簡単に使えるように定型化しています。ゲーム起動後は「first.ks」から始まりますが、少し追ってみると `@eval exp="biscrat_handler.callExtraConductor('title.ks')"` というのがあります。title.ks の最初に call タグで飛んでいる感じだと思ってください。SDK のシステムスクリプトでは `exp=biscrat_handler.~` というのを多用します。KAG で少し tjs の機能を使っていた方は `exp=kag.~` をよく使ったと思いますが、それと同じようなものです。

title.ks の方を見てみると「状態の初期化」→「タイトル画面の表示」という流れになっています。初期化の方はそのままにしておいてください。ゲームエンディング後にタイトル画面に戻ってきて困らないように色々と初期化しています。

思い出しましたが、エンディング後にタイトルに戻ってくる際には `@eval exp="biscrat_handler.funcObject.goToTitle()"` で戻ってきてください。`next` タグなどで直接 title.ks に飛んでくると困ります。

first.ks の初期化で、`@position` については好きなように直してください。本編では position タグ自体の説明のために 01.ks の冒頭でやっていますが、title.ks で初期化と一緒にメッセージレイヤの設定もしてしまったほうが便利だと思います。

KAGEX ではメッセージレイヤにシステムボタンを設置できます。メッセージレイヤでスキップやクイックセーブができると便利ですよね。sysbutton タグを使ってそれができます。

```
@sysbutton normal=スキップ _normal over=スキップ _over on=スキップ _on  
x=100 y=40 exp=biscrat_handler.funcObject.switchSkip() nostable
```

button タグによるボタンと同じく、システムボタンはメッセージレイヤ上に配置されます。normal、over、on 属性にボタン画像を指定します。x,y 属性でメッセージレイヤ上の位置を指定します。locate タグは使わずに属性として使えます。nostable 属性は、true を指定すると文字を表示している最中などにもボタンがクリックできるようになります。storage 属性と target 属性でジャンプ先を指定することはできないので、exp 属性にクリック時に実行する TJS 式を指定します。ここに書く TJS スク

リプトは準備してあります。

biscrat_handler.funcObject.switchSkip() でスキップの開始、*biscrat_handler.funcObject.switchAutoMode()* でオートモード開始、*biscrat_handler.funcObject.switchMessage()* でメッセージレイヤの非表示、*biscrat_handler.funcObject.switchHistory()* で履歴の表示、*biscrat_handler.funcObject.goBackHistory()* で通過記録をたどる、*biscrat_handler.funcObject.exit()* で終了、*biscrat_handler.funcObject.quickSave()* でクイックセーブ、*biscrat_handler.funcObject.quickLoad()* でクイックロード、*biscrat_handler.funcObject.playCurrentVoice()* で現在のボイスの再生、*biscrat_handler.funcObject.goToSave()* でセーブ画面へ、*biscrat_handler.funcObject.goToLoad()* でロード画面へ、*biscrat_handler.funcObject.goToOption()* でオプション画面へ、*biscrat_handler.funcObject.goToTitle()* でタイトル画面へ、の機能が使えます。

これだけでも一般的なシステムボタンは作れると思います。これらへんのサンプルは、SDK公開には間に合わないと思いますが、後ほど公開するので参考にしてください。*title.ks* の *@position* と *@syncmsg* の間に *sysbutton* タグで追加してしまうのがお勧めです。

syncmsg タグはメッセージレイヤのみ表ページから裏ページへのコピーを行います。*position* などでメッセージレイヤの設定をした場合は、裏ページも更新しないとたまにバグるので *@syncmsg* はメッセージレイヤの設定をした後にセットとして書いておくのを推奨します。

実際のタイトル画面は * タイトル画面表示のところから書いてください。メッセージレイヤの 1 番あたりに、*position*、*button*、*link* タグを使ってタイトル画面を表示されるのを想定しています。これらへんは KAG と変わらないです。初期状態では画面を表示せずにすぐに *start ラベルに飛んで、そこからゲーム本編の 01.ks に飛んでいます。

ゲーム本編にジャンプするときは、*next* タグではなく *return* タグを使ってください。タイトル画面は *call* タグのようなもので飛んできています。これはセーブ / ロード画面やオプション画面を作る際にも同じです。*call* から戻るために *return* を使います。

そのセーブ、ロード、オプション画面ですが、「sysscn」フォルダの「system.ks」に作ることになっています。それぞれ *save、*load、*option ラベルで表示させます。タイトル画面と同じように、*button* タグや *slider* タグなどを使って頑張って表示させることになります。KAGEX だとシステム画面は TJS を使って書くのがお勧めです。*slider* などが最初から使えるので KAG よりはマシですが、タグを使ってシステム画

面を構築するのは同じくらいしんどいです。これらへんのサンプルも後で公開できればするかもしれません。

一番のお勧めは自分に発注しちゃうことですねー(笑)システム画面の組み込みやKAGEX導入のサポートくらいなら同人で1万円程度でも引き受けます。シナリオスクリプト部分は誰でもできるレベルで簡単なので、システム部分のみ人に任せるというのはアリだと思います。

■リリース

リリース時はKAGと特に変わりません。リリーサ使ってまとめてしまえばOKです。

■メニューとか

デフォルトのKAGEXと比べてもゲーム内メニューは拡張しています。表示/非表示あたりもツールで切り替えできるようになる予定なのでもう少し待ってください。

■ボイス

ちゃんとシナリオ順に番号振って、セリフごとに自動的に再生させるのがお勧めです。台本出力の時にどうせ番号ふるし……ということなら毎回voice属性でちゃんと番号振ってもいいかもしれません。今までKAGEX使っていて1度だけボイス番号がずれるバグに遭遇しました。何が起きたのかわからないですが多分2度と再現できないので気にしてません。選択肢などで分岐する場合はボイス番号がずれると思うので指定忘れないようにしてください。ボイスのデバッグ時はメニューにある倍速再生がかなり便利です。しっかり聞き取れなくても何かおかしいってのは早く再生してもわかるものです。1つ1つちゃんと聞く時間があればそれでいいんですが。

■動画再生

KAGだと何かタグがたくさんあってめんどくせえのですが、KAGEXでは環境レイヤのmovie属性使ってしまうと簡単です。`@layer name= 動画 movie=movie.mpg`のようになります。再生待ちには`@wv`を使います。

```
@layer name=動画 movie=movie.mpg  
@wv  
@動画 delete
```

これだけです。

■ウェイト

作者がゲーム中に待たされるのが大嫌いなので本文中で使ってませ

んが KAG の @wait も普通につかえます。それに加えて、大抵の同期動作のタグでは wait 属性が使えます。属性値に時間を指定すると、同期動作終了後にその時間だけウェイトが入るようになります。@街昼 fade=1000 sync wait=1000 では、フェード終了後にさらに1秒ウェイトされることになります。wait 属性をつけると sync 属性も自動的に入るのでそちらは省略してしまっても大丈夫です。@街昼 fade=1000 と @wait time=1000 の 2 つに分けると飛ばすのに 2 クリック必要ですが、wait 属性で 1 つにまとめると 1 クリックで飛ばせます。まとめてあげてください。

■[タグ]について

本書では [] で囲んだタグを一切使っていません。KAGEX では改行や空行での自動改行や自動ページ送りがあるので [] を使うとちょっとめんどくさいことになる場合があります。@ で始まるコマンド行ならその心配が無いので自分の場合はこちらに統一しています。

■ラインモードについて

本書では、テキスト中の改行はそのまま改行、空行はページ送りとなっています。ここらへんの動作は「ラインモード」として設定できるのですが、細かいことはツールに付属する説明書みてください。作品によっては変える必要があるかもしれません。改行ごとにクリック待ちをさせるラインモードなどもあるのですが、全く推奨しません。できるだけクリックさせて欲しくないです。

■テキスト全画面表示について

KAGEX はテキストを画面全体に表示するゲームにはあまり向いていません。まあできないことはないですが、名前欄の機能は使えなくなってしまうのではないかどうか。

■トランジション

KAG の trans タグは使わないでください。管理が面倒になるだけです。

■時間切れ

作者のブログ (<http://kasekey.blog101.fc2.com/>) でもいくらか KAGEX 関連の記事があります。そちらもウォッチしてみると参考になるかもしれません。メール (sakano@biscrat.com) で質問も一応は受け付けてます。Twitter(<https://twitter.com/kasekey>) でもどうぞ。

吉里吉里 /KAGEX 講座
～デジタルノベル製作～

発行日 2011年12月31日 初版発行
著者 sakano

Email sakano@biscrat.com
Tel 080-5095-02037
Web <http://biscrat.com/>
Twitter <http://twitter.com/kasekey/>

ご意見ご感想お待ちしております。

Copyright (C) 2011 Biscrat.