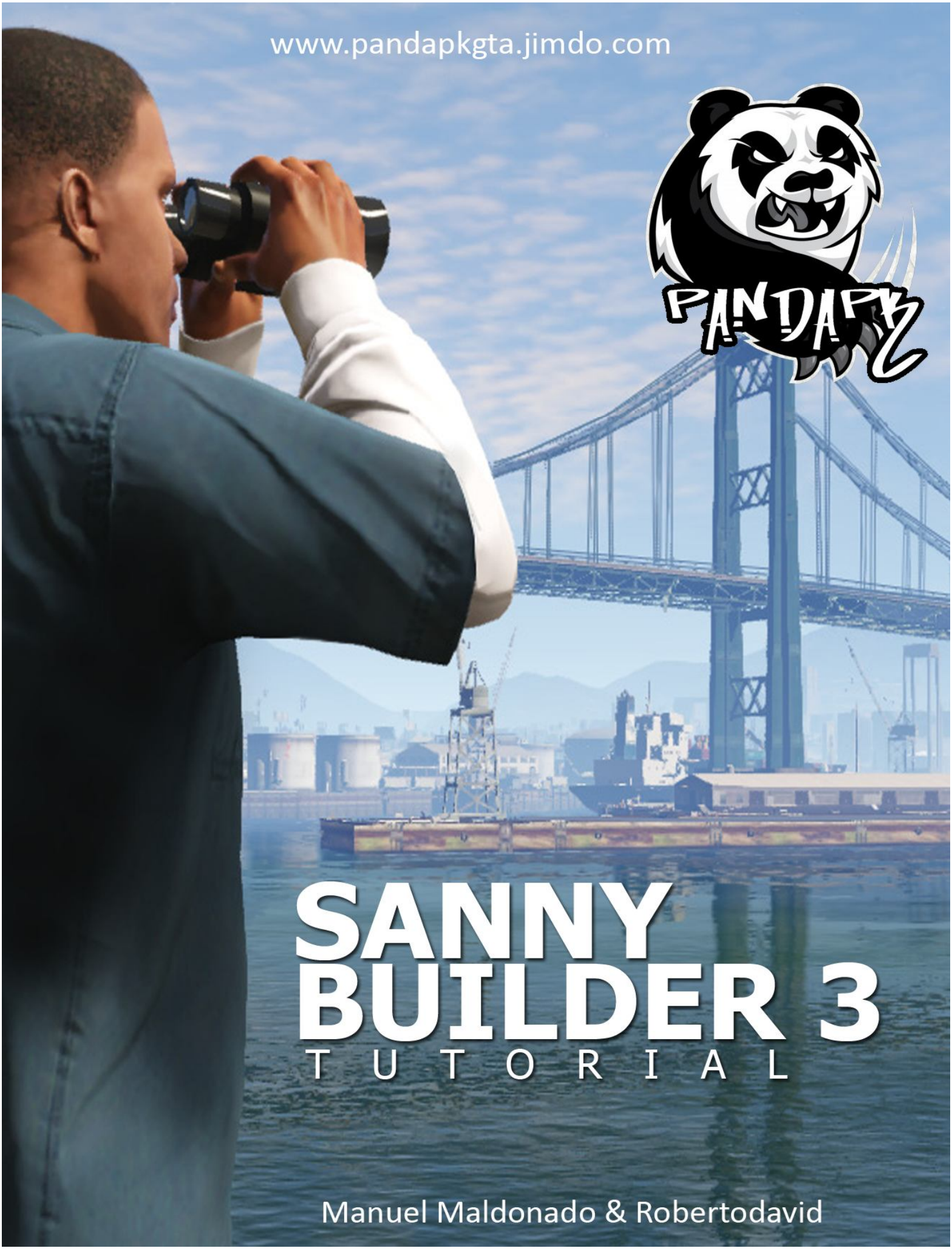
www.pandapkgta.jimdo.com
PANDAPK
SANNY
BUILDER 3
TUTORIAL
Manuel Maldonado & Robertodavid

1 COMO CODIFICAR Y USAR SANNY BUILDER 3

CONDIFICACION

Hay mucha gente que quiere aprender de codificación, pero cuando ven el código, se dan por vencidos. Este es un tutorial grande que caminar a través de las opciones básicas de SCM-codificación. Puedo prometer que usted es capaz de crear una misión sencilla cuando se trabajó a través de este tutorial. La mejor manera de aprender con este tutorial, es tratar cada guion completo, por lo que veremos lo que pasa dentro del juego. De esta manera usted aprenderá más rápido y mejor, porque se puede visualizar en su mente.

La diferencia entre este tutorial y la mayoría de los otros, es que este tutorial está escrito desde otro punto de vista. Que contiene (en este momento) 12 tutoriales cortos, que se puede recoger por separado. En cada tutorial, aprenderás otro aspecto de la codificación. Los 3 primeros tutoriales no son las cosas dentro del juego, se trata de la codificación en sí y sobre las estructuras. Usted tiene que conocer de primera, pero esto no será tan agradable como los tutoriales que siguen. A continuación, usted aprenderá cosas como la creación de los actores y los coches, los fundamentos de una misión. Casi todos los tutorial terminarán con un archivo despojado y el código, de modo que usted puede utilizar en SannyBuilder para ver lo que había aprendido. También puede utilizar esto como base para sus propios experimentos para ver qué hacen otros códigos de operación.

Vamos a utilizar SannyBuilder, porque el aprendizaje de código es más fácil con esta herramienta.

Para estos tutoriales que necesitas:
SannyBuilder (Download)
A "despojado" de código (SannyBuilder carpeta> Datos> SA> Stripped)
Grand Theft Auto: San Andreas
Ped Editor (Download)
San Andreas Administrador Place (Descarga)
Contenido:

Escenas
Enlaces:
Reactiontopic - para todas las respuestas a este tutorial.
Misión de codificación para Pro
SMC Codificación FAQ
Misión de codificación subforo
CLEO script Tutorial
Codificación Biblia Parte I
Codificación de la Biblia Parte II
Introducción

Contenido:

Introducción

Este tutorial es sólo para personas que quieren aprender de codificación. Cuando usted desea aprender de codificación, ¿dónde hay que empezar? Usted no puede ver el bosque por los árboles. En este tutorial, voy a ayudarle con estos primeros pasos. Las medidas que la mayoría de la gente piensa que son difíciles, por lo que renuncian. Todos los que ponen esfuerzo en él, puede aprender de codificación! Pero no creo: yo quiero aprender de codificación, así que mañana voy a hacer mi primera misión. Esto por supuesto no es la forma correcta de hacerlo. Aprender a código lleva su tiempo, al igual que cuando se quiere aprender a usar otros programas, como Photoshop o 3dsmax.

¿Qué necesito?
Algunas personas se pregunta qué se necesita el código:
Un programa para editar main.scm
Un editor de GXT (GXT Editor SA)
San Andreas Administrador de lugar (no de coordenadas, pero para teletransportarse)
Editor de mapas
Ped Editor
Tiempo / Esfuerzo
Programas

Función MissionBuilder SannyBuilder el Bloc de notas
Edición de sí sí sí (cuando ya descompilado)
Compilar sí sí no
Descompilar Sí Sí No
Actualizaciones no sí no
Pequeños extras Sí No
Creador Barton Waterduck Seemann Microsoft
Descargar Klik Klik -
Adicionales en SannyBuilder decir cosas tales como: CLEO3, construido en coordenadas-herramienta, herramienta de código de operación de búsqueda, muchos de teclas de acceso rápido, los textos de

color, etcétera. Aconsejo a todo el mundo que empieza a aprender el código, utilizar SannyBuilder.

SMC Codificación
Estructuras más importantes!
Por supuesto, todo el mundo quiere escribir el código de inmediato, por lo que las misiones u otras cosas. Yo aconsejo, porque la primera vez que aprenden las estructuras de la SCM-codificación, que será más fácil a la escritura después. Se puede comparar esto con la escuela: primero tienes que aprender las cosas (que es menos agradable) por lo que conseguir un trabajo mejor en el futuro. Esta es la razón por la que se iniciará con las estructuras después de la introducción de dos tutoriales.

Codificación
Después de aprender las estructuras, no debe tratar de hacer una misión total todavía. Un mejor comienzo es generar algunas camionetas. Entonces usted puede tratar de generar un coche o un actor, que explicaremos en tutoriales. Para que aprendas centímetro a centímetro para hacer una pequeña misión, que también va a hacer en tutoriales. Por lo tanto, si usted quiere aprender a hacer una pequeña misión, usted tiene que seguir todos estos tutoriales y lo aprenderá. Después de realizar una pequeña misión, se puede extender, poco a poco, por lo que se aprende de codificación más avanzados.

CLEO
CLEO es una extensión de SMC scripting. Se agrega principalmente dos cosas: CLEO-scripts y nuevos códigos de operación. Yo aconsejo empezar con SMC en primer lugar. Cuando se puede codificar SMC, también puede escribió el código de CLEO. La otra forma es más difícil, a causa de algunas estructuras dentro de SMC, que no tiene en CLEO.
¿Pero cuáles son las ventajas de CLEO?. Con CLEO-scripts no tiene juego nuevo comienzo, que es muy útil para la mayoría de los mods pequeños. También hay algunos códigos de operación de gran valor añadido. También puede utilizar los códigos de operación en su nuevo mods SMC, pero los usuarios de su mod que también es necesario tener instalado CLEO. Usted puede encontrar todas las diferencias entre CLEO y SCM en los archivos de SannyBuilder ayuda: Ayuda> Contenido> CLEO 3 Library.

Antes de comenzar

Descompilar
¿Qué es?
Antes de editar el archivo main.scm, primero hay que descomponer. Esto significa que se convierta en lo humanamente legible, a transformar la extensión de. Scm a. Txt. Usted puede hacer esto en los programas como MisionBuilder y SannyBuilder, pero no en el bloc de notas. Una vez que tenga el archivo txt, también se puede abrir el bloc de notas o cualquier otro editor de texto. Esto no es muy útil, porque no hay extras en el bloc de notas (lo cual lo hace mucho más difícil).

Cómo descompilar
Debido a que estos tutoriales se acerca SannyBuilder, lo voy a explicar por SannyBilder.
Presione la tecla F5 o el icono de "descompilar" o "ejecutar"> "descompilar".

Ir a la carpeta: Archivos de programa> GTA San Andreas> Datos> Script y seleccione "principal" (un

archivo de SMC.). Abrirlo.

Cuando usted está haciendo por primera vez, aparecerá una pantalla como la de abajo. Seleccione la carpeta "GTA San Andreas". Entonces usted tiene que hacer el paso anterior otra vez.

Ahora va a obtener el texto, que puede editar. Cuando lo guarde, obtendrá un archivo txt.

Compilar
¿Qué es?
Esto es lo contrario de descompilar. Cuando se desea probar el código, hay que transformar de nuevo a la lengua que GTA SA se puede leer. Así que tienes que hacer un SMC. Otra vez. Usted puede hacerlo pulsando F6, pulse el icono de "compilar" o "Ejecutar"> "compilar".

¿Cuál es la diferencia entre compilar y copiar + compilar?
Usted puede optar por guardar el archivo txt que está en otra carpeta en el pc de la carpeta de datos original. Compilar significa que el archivo SMC se crea en la carpeta donde se guarda el archivo. Txt. Compilar + Copia significa que el archivo SMC se copia en la carpeta de datos después de la compilación. Por lo que el main.scm en la carpeta de datos será reemplazado. No hay ninguna diferencia cuando se está editando la main.txt que se encuentra en los "datos" carpeta.

SannyBuilder
Colores

En SannyBuilder algunas palabras marcadas:

Negro: a menudo se usan los comandos, como saltar, esperar, create_thread, fundido, repito, JF, etcétera. Usted no tiene que poner el código antes de que.
Verde: son las etiquetas, por ejemplo ": label" o "etiqueta @".
Azul: las variables tienen un color azul.
Red: los nombres de los objetos ejemplo, los textos y las discusiones son de color rojo.
Amarillo y azul: El amarillo es el "objeto", el azul es lo que se dice sobre el objeto. Es difícil de explicar, pero si nos fijamos en él, estoy seguro de que lo entenderá.
Brown: los nombres de los modelos.
Azul / gris (la luz): esto son los comentarios, no son leídos por el juego y son muy útiles como una nota. Se procedió con / /.
Funciones
Existen algunas funciones en SannyBuilder, que son muy útiles para su uso. Por supuesto, hay más funciones, pero estos son los más importantes.
Coordenadas gerente. Se puede encontrar en Herramientas> Herramientas Ide> Coordenadas Manager (tecla de acceso rápido: CTRL + ALT +1). Usted puede encontrar el X, Y y Z de coordenadas del lugar donde usted se encuentra en San Andreas. También puede encontrar el ángulo. No puede utilizar esta opción cuando SA no se está ejecutando, por supuesto.
Código de operación de búsqueda. Se puede encontrar: Herramientas> Ide Herramientas> Buscar Opcode (tecla de acceso rápido: CTRL + ALT +2). Con esta herramienta se pueden encontrar los códigos de operación muy sencilla.

La tecla F1: cuando typ una pequeña parte de un comando, sannybuilder a tratar de encontrar un comando que se utilizó en el main.scm original.
SMC documentación. Se puede encontrar en la ayuda del menú (tecla de acceso rápido: F12). Usted encontrará una base de datos útiles. Usted puede encontrar aquí, por ejemplo, el icono de los números.
Inicio SA: usted puede comenzar SA pulsando F8 o el icono de "Ejecutar San Andreas".
Missionbuilder> Sannybuilder convertidor: no funciona perfecto, pero usted lo puede encontrar en Herramientas-> menú Convertidor de código> MB -> SB.

Estructuras de codificación

Temas
El archivo main.scm es procesado por múltiples threads.This podría ser uno de los conceptos más difíciles de entender para alguien no familiarizado con la programación multi-threading. Así que no te desesperes si no lo consigue de inmediato, no estás solo.

Hasta ahora, hemos descrito el código que se ejecuta código de operación después de código de operación, a veces salta a otro lugar y continuar. Este es un proceso ejecutado por un pequeño hombre dentro de su computadora. Se pasa todo el día leyendo un código de operación, de ejecutarla, para determinar el código de operación junto a ejecutar, lo leyó y lo ejecuta, día a día de salida. Permite llamar a este pequeño "hilo".

Ahora la imagen que estos chicos tiene un hermano, que hace exactamente lo mismo. Por supuesto que no estamos buscando a los códigos de operación misma, al mismo tiempo, su hermano está trabajando en otra parte del código. Ambos están trabajando de forma independiente el uno del otro, la lectura de códigos de operación, la ejecución de ellos, la determinación de código de operación que viene, ...
¿Puede que la imagen?

Ahora la imagen que hay toda una familia de esos pequeños individuos, todos haciendo lo mismo en diferentes partes del código.

~~~~~ Bienvenido al mundo de la multi-threading. ~ ~ ~ ~

Ahora, ¿cómo manejar estos temas?
Al principio sólo hay un hilo que se empieza a ejecutar en el principio del archivo main.scm, llamado el principal hilo. A través de una serie de saltar la va a saltar las primeras tres secciones y llegar al principio del código de la sección. Allí se llevará a cabo un poco de inicialización, hasta que encuentra una serie de "create_thread" códigos de operación.

Código de operación que crea un nuevo "pequeño" (tema) que se inicia la ejecución de códigos de operación por debajo de la etiqueta. De esta manera, el hilo conductor, crea un ejército de hilos, cada uno trabajando en una parte específica del código, para realizar una tarea específica. Algunos de los temas del vivir de forma indefinida, sólo girando alrededor del mismo código. Otros hilos Sólo tiene que realizar una sola tarea una vez y matar a uno mismo al encontrarse con un "end_thread" código de operación.

Un elemento también puede matar a otro hilo, mediante el suministro de nombre de su víctima como un parámetro para el código de operación end_thread. Los hilos se le da un nombre, cuando se ejecuta un "name_thread" código de operación, que normalmente es uno de los códigos de operación primero que encuentran.

Create_thread pueden ser ejecutadas tanto de la parte principal, así como la parte MISIONES. El código que el hilo se ejecutará sin embargo, debe estar ubicado en la parte principal.
~~~~~

Misiones también se ejecutan como hilos, sino que se inician con "la misión de comenzar". Esto creará una clase especial de la OIF hilo que solo puede existir al mismo tiempo. Además, durante este hilo del motor comprueba continuamente si el jugador está roto o perdido. En este caso, el "pequeño" es instruido para llevar a cabo un "retorno" código de operación, lo que le hace saltar de nuevo a la última "gosub". Esta es la razón por las misiones deberían tener una estructura especial, en la que esto llevará a una misión fallida. Esta estructura se describe en uno de los tutoriales que vienen.

Etiquetas / saltar / gosub
Las etiquetas están marcados con un ":". Estos son los nombres que se asignan a algunos lugares en el código. Se utiliza saltar a una ubicación en el código. Usted puede hacer esto con "gosub" y "saltar".

CÓDIGO
```
: MAIN_4059
01B7: release_weather
salto @ MAIN_4075
```

Usted puede ver una etiqueta, llamada "MAIN_4059". Entonces usted puede ver un comando, con un código (01B7) llama código de operación. En la tercera línea se puede ver otro comando: "saltar". Con la "@..." después de que usted indica que el código tiene que saltar. Por lo que se están subiendo a ": MAIN_4075".

Gosub diferentes obras, sino que también salta a otra etiqueta, sino que se remonta a la ubicación justo después de la "GOSUB" cuando llega a un "retorno" al mando. Por ejemplo:

CÓDIGO
```
: MS_GAME_BEEFYBARON
$ ONMISSION = 1 / / valores enteros
00BA: text_styled 'BEEFY 1000 de 2 ms / / Beefy Baron
gosub @ SUB_FADE_500MS
start_mission 10 / / Beefy Baron
salto @ END_CASE_VIDEO_GAME
```

CÓDIGO
```
: SUB_FADE_500MS
0169: set_fade_color 0 0 0
fade 0 500

: LITCAS_282
si
desvanecimiento
else_jump @ LITCAS_306
esperar 0
salto @ LITCAS_282

: LITCAS_306
Volver
```

Por lo tanto, salta con un "gosub" a ": SUB_FADE_500MS". Cuando llega a la "vuelta" de comandos (bajo ": LITCAS_306" que se remonta a la gosub. Después del regreso, el código seguirá con "start_mis

Gosub se utiliza cuando se desea utilizar una pieza de código en múltiples lugares en el código.

Salto condicional
Cuando se utiliza un "salto condicional", el código va a hacer o no hacer el salto dependiendo de alguna condición. Puede utilizar tres anotaciones para esto, que todos tienen el mismo resultado:
"Else_jump @ boe"
"Jf @ boe"
"004D: jump_if_false @ boe"

CÓDIGO
: SWEET4_103
si
Model.Available (# GREENWOO)
else_jump @ SWEET4_152

En este ejemplo se comprueba si el modelo está disponible (el modelo de Greenwood, en este caso). Cuando esto no es cierto, que salta a la etiqueta ": SWEET4_152". Cuando esto es así, el código va más allá.

Con el salto condicional, se puede hacer un lazo. A continuación, el código de espera hasta que la condición es verdadera. ¡Tenga cuidado! Usted tiene que poner una espera en cada bucle! Cuando se le olvida, el juego se bloqueará, y, cuando no se ejecutan SA en una ventana, aunque no puede volver a su escritorio. La única solución es reiniciar el PC. Por lo tanto, mi consejo es ejecutar SA en una ventana en el testeo de código nuevo! Un ejemplo de un bucle es la siguiente:
CÓDIGO
: SWEET4_103
Espere 0 ms
si
Model.Available (# GREENWOO)
else_jump @ SWEET4_103

Como puede ver, el código permanece en este bucle, hasta que el modelo está disponible.

Si-entonces-finales estructura
Hay otras estructuras que se pueden utilizar para comprobar las condiciones. Por ejemplo, la estructura si-entonces-end. Por ejemplo:
CÓDIGO
si $ eleccion == 0
entonces
$ Dinero = 10
Final

Cuando la variable $ opción es igual a 0, entonces la variable $ dinero se convertirá en 10. Si $ elección no es igual a 0, no pasará nada. Tienes que poner un "final" para indicar que el código debe seguir cuando la condición no es cierto. Usted puede agregar más cosas entre el "entonces" y "end" que sólo se ejecutará cuando la condición es verdadera.

Repetir esperar hasta que la estructura
CÓDIGO

repetir
esperar a 0 ms
hasta 03D0: wav 3 cargado

Este es un ejemplo de una estructura de repetición de espera hasta que. Se trata de un bucle, así que tiene que haber una espera en él. Este código significa: hacer todo lo que entre la repetición y hasta, hasta que la condición es verdadera. Entre la repetición y hasta que no puede haber también otras cosas, que se ejecutará una y otra vez hasta que la condición es verdadera. En lugar de utilizar repetidas de espera hasta que, también se puede escribir como un salto condicional (esto se comportan exactamente lo mismo!):
CÓDIGO
: Etiqueta
esperar a 0 ms
si
03D0: wav 3 cargado
else_jump @ etiqueta

Números
Hay dos tipos de números:
Los valores enteros: 1 2 3
Valores de punto flotante: 1.5345 3.4

Valores de punto flotante se utilizará como coordenadas, ángulos o cosas como la velocidad.

Los valores enteros se utilizan para la mayoría de otras cosas. La ventaja de los valores enteros, es que la computadora puede calcular fácilmente con estos. Estos números son, por ejemplo, utilizada por un período de espera.

Los valores enteros y los valores de punto flotante no se pueden mezclar, ya que el código no va a funcionar.

CÓDIGO
0004: $ abc = 5000;; valores enteros
0084: $ def = $ abc;; valores enteros y se ocupa de

La diferencia es que los primeros asignan un valor a una variable global. En este caso, de 5000 a la variable global $ abc. Entonces usted dice que $ def tiene que obtener el mismo valor que $ abc. Por lo que este código significa que: $ $ abc def = = 5000 (pero no se puede utilizar esto en el main.scm, no tienes que dividir)

Variables
Hay dos tipos de variables: globales y locales. Las variables globales se indican con un $ (por ejemplo: $ abc), locales se indican con un @ al final (por ejemplo: 2 @). La variable local-sólo los nombres se compone de un número, una variable global tiene que constar de los caracteres y digts. Cuando una variable global sólo se compone de números, se maneja de manera diferente por el compilador. Esto puede conducir a resultados inesperados, por lo que debe asegurarse de que su variablenambles contienen al menos un carácter. La diferencia entre la variable local y global es el uso de un mundial a través de todo el guión. Una variable local se aplica sólo dentro de un hilo. Al asignar un valor a un @, y

desea utilizar ese número en otro hilo, que no será disponible, hay que asignarle ese valor nuevo. El valor del otro hilo no se aplica más.

Calcular con variables
A veces hay que calcular con variables. Hay algunos códigos de operación de este. Atención: hay diferentes códigos de operación de entero-los valores y los valores de punto flotante, y hay diferentes códigos de operación para las variables globales o locales. Puede sumar, restar, multiplicar y dividir. Algunos ejemplos:

CÓDIGO
0008: $ variable + = 1

CÓDIGO
000A: 3 @ + = 3000

CÓDIGO
000B: 6 @ + = 0.1

CÓDIGO
0009: $ variable + = 1.741

CÓDIGO
0058: $ variable + = $ variable23 / / (int)

Tenga cuidado de no confundir estos dos códigos de operación:
CÓDIGO
0038: $ variable == 1

Con este código de operación, se comprueba si $ 672 es igual a 1.

CÓDIGO
0004: $ var = 0

Con este código de operación que se establece la variable global $ var a 1.

Actores

Hay dos tipos de actores: "normal" los actores y los actores "especial". Los actores normales son más fáciles de hacer, porque tienen que ser cargados al igual que los modelos normales (como las armas). Los actores especiales son un poco más difícil, pero te lo explicaré más tarde. Vamos a empezar con un "pelado" de código, que es un código que está limpio. Lo único que podemos hacer es caminar / conducir alrededor. Es perfecto para un nuevo mod. Un actor es un personaje en el juego, como dulce, pero también los peatones al azar. En primer lugar vamos a generar un actor normal. Tienes que hacerlo en tres pasos:

- Los modelos de carga actor (y comprobar si está cargado)
- Spawn actores
- Descarga de modelos

Modelos de carga actor (y comprobar si está cargado):
CÓDIGO
/ / Cargar los modelos
0247: load_model # BFYST
038B: load_requested_models

Creo que esta parte del código habla por sí mismo, pero lo explicaré en breve. Al típico / /, es un comentario, como fue dicho en tutoriales anteriores. Esto es sólo una nota para el codificador. En la línea "load_model", que tiene que decir cuál es el modelo que desee utilizar. Si usted está buscando un ped específico, usted puede encontrar el nombre utilizando el editor de ped. Que he elegido para BFYST, pero también hay otras posibilidades. Tienes que escribir la tercera línea, porque entonces los modelos anteriores en realidad van a cargar.

Entonces usted tiene que comprobar si el modelo se carga:
CÓDIGO
: MODEL_LOAD
esperar a 0 ms
si
8248: No disponible el modelo # BFYST
else_jump @ MODEL_SPAWN
salto @ MODEL_LOAD

: MODEL_SPAWN

Usted tiene que hacer una nueva etiqueta, porque va a crear un bucle. Entonces usted sabe donde se puede saltar. No importa qué nombre darle a la etiqueta, lo llamé "MODEL_LOAD". Te aconsejo que le dan un nombre claro, porque es más fácil encontrarlo cuando se tiene un gran mod.
Cuando has seguido los tutoriales anteriores, usted sabe lo que el salto, si es falsa (o else_jump) es la estructura. Este es un ejemplo de su uso. Si el modelo "BFYST" no está disponible, luego salta a MODEL_LOAD. Si está disponible, el salto a: MODEL_SPAWN.

El siguiente código es el mismo:
CÓDIGO
: MODEL_LOAD
esperar a 0 ms
si
0248: modelo de 15 @ disponibles
else_jump @ MODEL_LOAD

Generar Actor
CÓDIGO
: MODEL_SPAWN
10 @ = Actor.Create (CIVFEMALE, # BFYST, 0.0, 0.0, 0.0)
Actor.Angle (10 @) = 318.9705

El 10 @ es una variable local, no importa qué número está utilizando. Tiene que ser en los 32, debido a que es el máximo dentro de un hilo (cuando se utiliza un "oficial" de la estructura de la misión, que puede

ser más de 33). A continuación, vamos a asignar a la variable un valor, en este caso un actor. Así que le damos al actor, en este caso, el nombre de "10 @". Actor.Create es el comando, creo que habla por sí mismo. Después de eso, verá cinco parámetros. En primer lugar el tipo de agregados, CIVFEMALE. A continuación, el nombre del modelo y después de que hay tres coordenadas: x, y, z. Como ya he dicho en tutoriales anteriores, existe una herramienta integrada de coordenadas en SannyBuilder, puede utilizarlo para determinar las coordenadas como estos.
El ángulo de actor es la dirección de los PED. 0.0 es hacia el oeste. Así que si quieres que se pare al este, tienes que darle un ángulo de 180.0. También puede utilizar la herramienta de coordenadas para esto.

Modelo de descarga
Por último hay que descargar el modelo:

CÓDIGO
0249: release_model # BFYST

¡Cuidado! Release_model no significa que el actor va a desaparecer! Al igual que la carga del actor no significa que generar el actor. Usted tiene que usar otro código de operación para eliminar el actor (por ejemplo, 009B: destroy_actor 18 @)

Por lo tanto el código completo de un actor se generan:
CÓDIGO
/ / Cargar los modelos
0247: load_model # BFYST
038B: load_requested_models

: MODEL_LOAD
00D6: si
8248: No disponible el modelo # BFYST
004D: jump_if_false @ MODEL_SPAWN
0001: esperar a 0 ms
0002: Salto @ MODEL_LOAD

: MODEL_SPAWN
10 @ = Actor.Create (CIVFEMALE, # BFYST, 0.0, 0.0, 0.0)

/ / Aquí se hace lo que quiere hacer con el actor.

0249: release_model # BFYST

Con el Editor de Ped usted puede encontrar su ped-modelo, que también tendrá un punto de vista de los PED (cómo se ve dentro del juego). Usted verá el nombre del PED y el grupo / tipo. Así que para hacer un desove actor, es necesario SannyBuilder (para editar el archivo), herramienta de coordenadas (en SannyBuilder) y Editor Ped para buscar los nombres de los modelos de actor.

Actores especiales

Actores especiales son un poco más difícil. Usted tiene 10 "slots" para los modelos de actores especiales. Se puede comparar esto con el 8 ranuras en las que se puede guardar partida guardada: sólo puede tener 8 a la vez, y si quieres otra partida guardada, tiene que eliminar primero. Para ver una

lista de todos los agentes especiales, usted puede buscar en el archivo de ayuda: Ayuda> Contenido> Subvenciones> Documentación GTA SA> Actores Especiales.

Cuando usted entiende los actores normales, también se puede entender los agentes especiales. Parece una similar, pero con el código de un poco diferente. Una vez más, usted tiene que cargar el modelo, comprobar si el modelo está disponible, generar el actor y descargar el modelo.

Carga del modelo
Tomamos Catalina, por ejemplo. Verá, cuando se mira la documentación de SMC, que ha 'CAT' en el nombre. En primer lugar vamos a cargar el modelo:
CÓDIGO
/ / Cargar los modelos
023C: 'CAT' load_special_actor como 1 / / los modelos 290-299
038B: load_requested_models

Así que en este código, la carga Catalina actor especiales (CAT), en la ranura 1. Junto a éste, puede cargar hasta nueve otros.

Compruebe si está cargada:
CÓDIGO
: MODEL_LOAD
si
823D: no special_actor una carga
else_jump @ MODEL_SPAWN
esperar a 0 ms
salto @ MODEL_LOAD

: MODEL_SPAWN

Esto es casi el mismo que el "normal" actor. Sólo la segunda línea es diferente, la estructura es la misma: cuando el modelo de actor especial, no se carga, el salto a la MODEL_LOAD etiqueta. Si se carga, el salto a MODEL_SPAWN. El uno es en este caso el número de la ranura, en el que está cargado.

Desove del actor
CÓDIGO
: MODEL_SPAWN
009A: 10 @ = create_actor_pedtype CIVFEMALE modelo # SPECIAL01 a 0,0 0,0 0,0
0173: set_actor # SPECIAL01 Z_angle_to 0.0

Es exactamente el mismo que el "normal" los actores, excepto que los modelos se denominan # SPECIALxx. XX depende de la ranura que se utiliza. En este caso, es 01, porque era la ranura 1. Si se trata de la ranura 2, usted tiene que utilizar: # SPECIAL02.
CIVFEMALE es el pedtype, el ángulo es igual que los actores normal.

Descargue el actor
Por último hay que descargarlo:
CÓDIGO
0296: 1 unload_special_actor

Por lo tanto, descargar el modelo de actor especial, que está en la ranura 1.

Todo el código para generar un actor especial:
CÓDIGO
/ / Cargar los modelos
023C: 'CAT' load_special_actor como 1 / / los modelos 290-299
038B: load_requested_models

: MODEL_LOAD
00D6: si o
823D: no special_actor una carga
004D: jump_if_false @ MODEL_SPAWN
0001: esperar a 0 ms
0002: Salto @ MODEL_LOAD

: MODEL_SPAWN
009A: 10 @ = create_actor_pedtype CIVMALE modelo # SPECIAL01 a 0,0 0,0 0,0
0173: set_actor 10 @ Z_angle_to 0.0

/ / Aquí se hace lo que quiere hacer con el actor.

0296: 1 unload_special_actor

Al incluir un actor normal a un archivo despojado!

Puede dejar que el actor hacer muchas cosas, como entrar en un coche, o disparar a alguien. Pero primero usted tiene que aparecer, entonces usted puede agregar otras cosas. Es por ello que es esencial para saber esto. Espero que hayas aprendido algo de este tutorial, y si usted tiene alguna pregunta, hágamelo saber!

Dar armas a los actores

Introducción
En la segunda parte de este tutorial, vamos a extender el código de la parte 1. Vamos a darle al actor un arma, y pondré algunas otras propiedades. El propio jugador tiene un arma también.

Para hacerlo más divertido, el jugador y el actor tiene que matarse unos a otros. Si matas al actor, tendrá música, una misión de pasar el texto y el dólar 25000.

Vamos a empezar por dar al actor un arma.

Antes de dar un arma a un actor (el jugador también es actor!), Lo primero que tiene que cargar el arma-modelo. Al igual que hicimos en la parte 1 para los actores. Es exactamente lo mismo que la carga del modelo de "normal"-actor.

Este era el código para cargar un "normal", el actor:
CÓDIGO
/ / Cargar los modelos
0247: load_model # BFYST
038B: load_requested_models

: MODEL_LOAD

```
esperar a 0 ms
si
8248: No disponible el modelo # BFYST
else_jump @ MODEL_SPAWN
salto @ MODEL_LOAD

: MODEL_SPAWN
```

Vamos a ampliar esta parte del código mediante la carga de los modelos de derecho de tres armas: el desierto-águila, el lanzador de cohetes, y la M4.
En los archivos de ayuda de SannyBuilder que buscar los nombres y los números de modelo de las tres armas: Ayuda> SMC Documentación> GTA SA> Números de armas.

Allí encontramos los siguientes datos:
águila del desierto-24 # DESERT_EAGLE
lanzacohetes 35 # ROCKETLA
M4 31 # M4
Hacemos extensivo el código anterior con los nombres de modelo de las armas:
CÓDIGO

```
/ / Cargar los modelos
0247: load_model # BFYST
0247: load_model # rocketla
0247: load_model # desert_eagle
0247: load_model # m4
038B: load_requested_models

: MODEL_LOAD
esperar a 0 ms
si o
8248: No disponible el modelo # BFYST
8248: No disponible el modelo # rocketla
8248: No disponible el modelo # desert_eagle
8248: No disponible el modelo # m4
else_jump @ MODEL_SPAWN
salto @ MODEL_LOAD

: MODEL_SPAWN
```

Ahora, el código se mantendrá en el circuito, siempre y cuando ninguno de los modelos no se ha cargado todavía. Usted tiene que hacer "si" a "o si", porque no hay más que una condición. "Si o" significa que una de las condiciones tiene que ser verdad ", y si" significa que todas las condiciones tienen que ser verdad. Así que en este caso, el código de espera hasta los 4 modelos están cargados.

Ahora vamos a dar a las armas. Vamos a empezar por dar al jugador un lanzacohetes y un águila del desierto:
CÓDIGO

```
01B2: $ give_actor arma PLAYER_ACTOR 35 municiones 10 / / Cargar el modelo de arma antes de usar este
01B2: $ give_actor arma PLAYER_ACTOR 24 municiones 30000 / / Cargar el modelo de arma antes de usar este
01B9: set_actor $ PLAYER_ACTOR armed_weapon_to 24
```

Aquí se necesita el número de las armas que hemos encontrado en los archivos de ayuda. Vamos a dar al jugador una rocketlauncher (número 35) con 10 cohetes y un desierto-real (número 24) con 30.000 balas.
El jugador tiene dos armas, pero sólo puede usar una a la vez. Para definir lo que está en su mano, usted tiene que usar el "01B9" código de operación. En este caso el jugador tiene el desierto, el águila en sus manos.

Ahora vamos a dar a un actor en un arma:
CÓDIGO
01B2: give_actor 10 @ 31 armas munición 30000 / / Cargar el modelo de arma antes de usar este
01B9: set_actor 10 @ armed_weapon_to 31

Por lo que el actor recibe un M4 (número 31) con 30.000 balas. También estará en sus manos.

Ambos actores (Player y otros) tienen armas. Ahora vamos a dar al actor una misión: matar al jugador. También vamos a establecer algunas otras propiedades del actor.

CÓDIGO
05E2: AS_actor 10 @ kill_actor $ PLAYER_ACTOR

El actor (10 @) tiene que matar al jugador.

CÓDIGO
actor.WeaponAccuracy (10 @) = 90

La precisión de las armas de que el actor es de 90, por lo que el actor rodará más precisa a continuación, cuando no se utiliza este código de operación.

CÓDIGO
actor.Health (10 @) = 2000

El actor tiene un montón de salud, por lo que es más difícil de matar.

CÓDIGO
0350: toggle_actor 10 @ maintain_position_when_attacked 1

El actor de mantener su posición cuando es atacado. De lo contrario, es posible que se escapa.

Ahora tenemos que el código de la final de la pelea. Tenemos que comprobar si el actor está muerto:
CÓDIGO
repetir
esperar a 0 ms
hasta actor.Dead (10 @)

Finalmente tenemos que establecer un sonido, un texto y una recompensa:
CÓDIGO
0394: 1 play_music
01E3: "M_PASS 'show_text_1number_styled GXT número 10000 tiempo estilo 5000 1 / / MISIÓN PASADO ~ n ~ ~ w ~ $ ~ 1 ~!
Player.Money ($ PLAYER_CHAR) + 25.000 =

La primera línea es para el sonido, el segundo es para el texto. Esto mostrará después de que el actor está muerto.
Vamos a dar una recompensa de $ 250,000.

Entonces tenemos que descargar los modelos, al igual que hemos aprendido en tutoriales anteriores:
CÓDIGO
0249: release_model # BFYST
0249: release_model # rocketla
0249: release_model # desert_eagle
0249: release_model # m4