Building a Basic DA:O Module

By Tom 'Bug.Bear' Smallwood
Version 0.7

Table of Contents

Part 1: Getting Started

Create a New Module

A module is a database containing all the elements used in the adventure, including levels, areas, characters, scripts, plots, cut-scenes and more. To create a new database, open the tool set and, from the top menu, click on [File], [Manage Modules] and then the [New] button.

You'll get window similar to Figure 1. This contains information specific to your module. There are only a few fields we care about right now, but feel free to look at the list to get an idea of what some of the options are.

The first field we need to set is the Name field, which is the name of your module. In this field, type *Rescue the Princess*. This is a required field.

The second field we need to enter is the UID. This is the name the game will use internally for the module. It may not contain spaces or other special characters. I normally start my modules with my initials, followed by an underscore and then a descriptor. In this case, I will use *TRS_Rescue*.

The final field we want check is the priority field. This field sets the precedence this module will take over other modules with resources named the same. All we really need to worry about is that the priority is greater then zero, which it should be by default. If it isn't, change it to 100, otherwise leave it alone.

We'll come back later and look at more of these fields, but for now, click [OK], then click [Open] so we can start working on the module.

You should now be looking at a mostly blank screen with a palette window and object inspector on the right hand side.

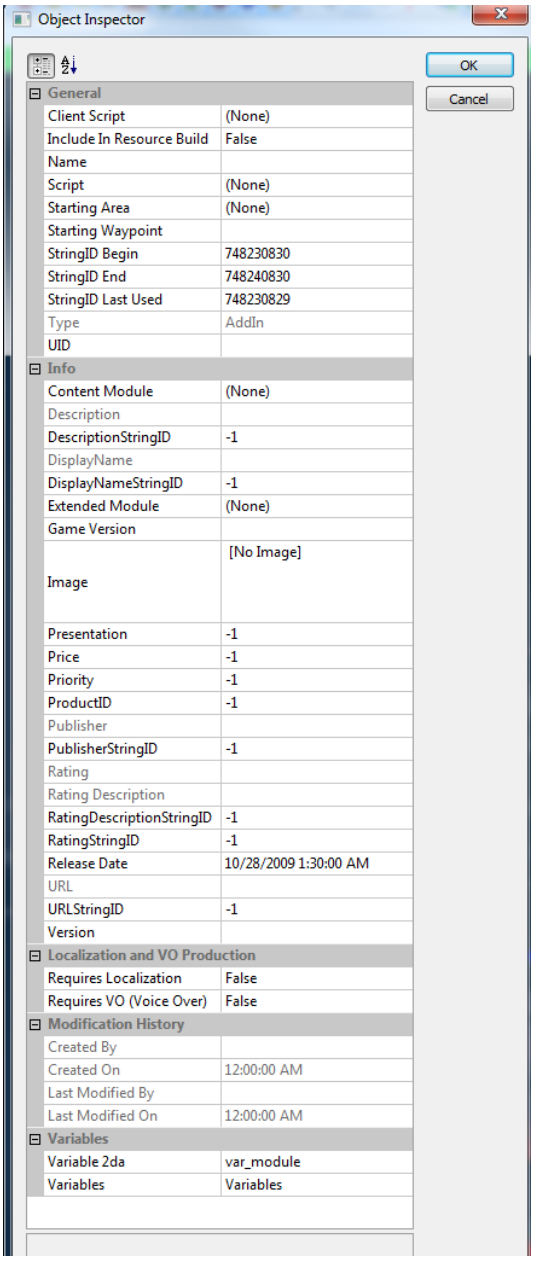


Figure 1: Module Object Inspector

Create An Area

Areas are the heart of a module where everything takes place. A completed area will usually consist of creatures, items, ambient sounds, triggers, scripts, NPC's, a level and more. Create a new area by going

to the menu and choosing [File][New][Area].

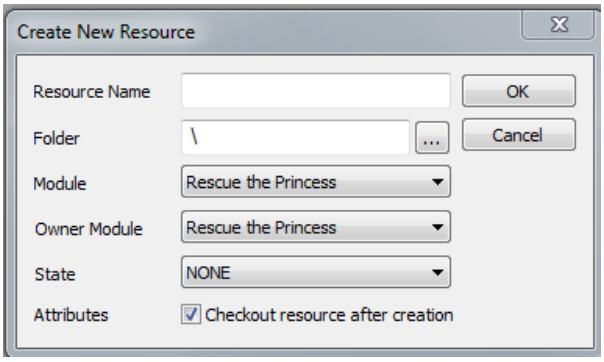


Figure 2: Create New Area Resource

Enter a name for the area in the Create New Resource Box. Resource Names may not contain spaces. The first area for our module will be the place the player will get the quest, in this case the castle. Change the Resource Name to *castle* and click [OK].

You'll now be looking at a black screen with a tree view in the left window. Select the area by clicking on word *castle* at the top of the tree view. The object inspector on the right will show you the properties of the area.

Now we need to load a level into the area. Levels are static, meaning you wont be able to change them in the area editor or the game. They can only be changed in the level editor. Levels include walls, props, ground, water, lighting and special effects.

For our purposes, we'll use a pre-built level that came with Dragon Age. Keep in mind you can create your own custom levels in the level editor. With the c*astle* area selected, go to the object inspector in the bottom right corner of the screen and click in the *area layout* field. On the right hand side of the field, find the browse button (...) and click it to get a list of available levels. We're going to use Den001d, so select it from the list and click [OK] to open it. You should now be looking at small level containing the inside our castle.

Read the gray side bar for directions on how to move around in the area viewer. Pan around and look at the level until you get comfortable navigating through the area.

Moving Around in the Area View

You can pan around the area by holding the ctrl key and the left mouse button. To rotate the view, use the ctrl key and the right mouse button. Alternatively, the middle mouse button will rotate the view. Finally, the mouse wheel will let you zoom in and out.

You can also use the number pad to move the camera. The + and – keys will zoom the view in and out. The 5 key will reset the camera to it's initial position.

If you have an object selected, such as a way point, you can zoom to that object by clicking on the *Zoom To Object* Button ( ). You can set an object to the center of rotation by clicking the *Focus on Object* button ( ).

Way Points and Start Points

Way points are are used to mark specific locations in an area for thinks like setting guard patrol points, positioning spawn locations and marking notes on the map. Way points have an X, Y, and Z coordinate marking their location as well as a facing.

The first thing we'll use a way point for is setting the starting location for the player. To create a way point, first reset the camera view by pressing 5 on the number pad, then go to the [Edit] menu and select [Insert Waypoint]. A way point will appear under the mouse cursor. Click anywhere on the blue carpet to place our start position. You will see a blue flag surrounded by a yellow-outlined box. The yellow box shows what the currently selected object is. This object is also the object that appears in the object inspector.

Now that we have a way point, we need to define it as our start point for the module. With the way point still selected, go to the object inspector and change both the Name field and the Tag field to *Start*. Also, change the colour field to start (this isn't required, but makes it easier to quickly identify your start point).

You can move and rotate this way point using the object manipulator as outlined in the gray side bar on the right. Rotate it 90 degrees so it's facing the end of the room by press the *3 Axis Rotation* button (or press '*e*' on the keyboard). Move the mouse over the blue ring around the start point until the ring turns yellow and, while holding down the left button, move the mouse to rotate the point until it's facing the stairs to the right. When the player begins playing the level, they'll be facing the same direction as the start point.

Next, we need to tell the module that this is the start point. Go to the module properties [File][Manage Modules] and open the *Rescue the Princess* module again. This time, we're going to change both the Starting Area and Starting Waypoint properties. After you click in the Starting Area field, click on the browse button (...) and you should see a list of your available areas. Select c*astle* and then [OK].

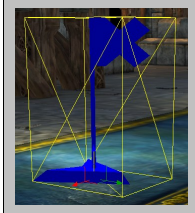
With the start area set to *castle*, the Starting Waypoint field will have a drop-down listing all of the way points in the area. Choose *start* from the pull down list. Click [OK] and then [Close].

At this point, you have a playable module. Go ahead and give it a try to make sure everything is working. Save your area and then go to [Tools][Export][Export with Dependent Resources]. A window will open and you'll get several informational messages in the log window. If everything worked correctly, the last line of the log window will give you an export summary and tell you how many export requests you had, how many succeeded, how many were skipped and how many failed. You should have 118 successes and no failures.

You can now run the game. From inside the game, on the main menu, choose Other Campaigns. Rescue the Princess should be one of your options. Select it and try it out. You don't need to close the tool set to try the module.

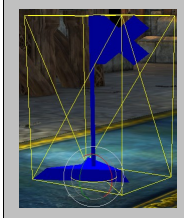
As you explore the area, you'll notice openings where doors should be and, if you walk through them, you'll end up outside the area. We'll fix that by blocking those openings with doors. Once you're done exploring, close the game and go back to the area editor. If the area isn't open, you can open it buy going to the palette and clicking the area button ( ),

Moving and Rotating Objects



To move the currently selected object, you can click the *3 Axis Movement* button on the toolbar or press Q on the keyboard to get a movement manipulator. This manipulator has 3 arrows, a red arrow on the X axis, a green arrow on the Y axis and a blue arrow on the Z axis. To move an object along only one axis, move the mouse over the arrow for the direction you want to move the object until it turns yellow. While holding the mouse button down, drag the object in the direction you want to move it.

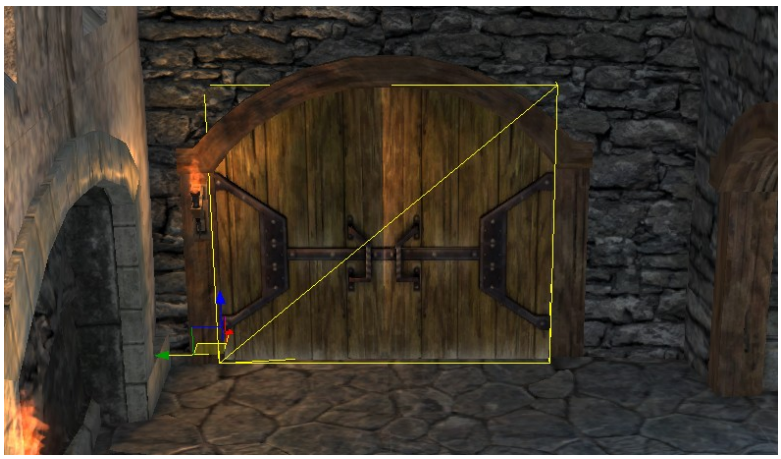
You should also notice that there are 3 squares near the center of the manipulator. If you move the mouse over one of these squares, the square and the two arrows it is connected to will highlight. If you drag the square, you will be able to move the object in two directions without moving it in the third.



You can rotate an object by selecting the 3 Axis Rotation button or by pressing the E key on the keyboard. You will see three circles forming a small sphere and a bigger circle around them. Similar to the movement arrows, each circle has a color and allows you to rotate the abject around the associated axis. The large, out circle lets you rotate the object around the current view.

then double click on *castle* to open it.

Each button on the palette represents a different type of resource available to add to our module. If you mouse over them, you'll see what each icon represents. To block the entries, we want to use a placeable object, so click on the placeable icon ( ). This will give us a nested tree list of placeable objects we can choose from.



Open up the *global* tree by clicking on the + symbol and then open up doors. You'll see a list of several door options. Pick the one labeled *genip_door_fer_dbl* (a generic double door). After clicking on it in the palette window, click in the view window to place it. Using the move and rotate manipulators, move the door over one of the openings in a hall which leads out of the area into the black space (refer to the gray side bar on the previous page to see how to move and rotate objects). If you have problems getting the rotation of the door perfect, you can always type in exact number for the door's orientation property (90,0,0) in the object inspector. Also, set the door's *interactive* property to false. This will keep the player from being able to open the door and leaving the area.

Now we need three more doors. Rather then pulling them off the palette, you can copy and paste them the doors from the first one. This will make sure the new doors have all the same properties we set on the first one. With the first door still selected, press ctrl-c to copy it. A new copy of the door will appear under the cursor. Pan the area view over to the next doorway and pace the new door so it blocks the entry. Do this again for the third and forth doors.

We're going to use the same placeable for the other four openings (the ones that lead from the side halls to the main room). Make a new copy of a door and move it to one of the openings, rotating it into position. Change the *interactive* value of the selected door to true. This will allow the player to open and close the door. Repeat this for the other 3 openings into the halls. We'll add the door at the front in the transition section below.

Creating a Quest (Plot)

Now that an area is set up, we are going to create a quest or, as the game calls it, a plot. Our plot is to rescue the princess and will have five steps required to complete it. To create a new plot, go to [File] [New][Plot]. In the resource window, enter *rescue_princess_plot* as the resource name and click [OK].

You'll now be looking at the plot builder. Click [Insert Main Flag] from the tool bar to insert the first step of the plot. Insert 4 more flags, one for each plot step. Flags have two states they can be in, either true (set) or false (not set). By default, each flag will have a value of false. When the step is completed,

we'll set the associated flag to true. Before we go any further, lets rename the flags to something a little more readable. Click on Flag_0 and change the text to *Quest_Accepted*. Change Flag_1 to *Talk_to_jack*, Flag_2 to *Key_Found*, Flag_3 to W*izard_dead*, and Flag_4 to *Princess_Rescued*.

Next, we need to create journal entries for our quest. Select the *Quest_Accepted* flag and, in the Journal Text field at the bottom of the screen, enter the first line of the plot from the gray side box (*Get Jack to help you rescue the princess.*). Do this for all 5 lines. This is the text that will appear in the player's journal. After all the journal entries have been entered, add one more flag and call it *Quest_Complete*. There's no journal entry for it, but change its '*Final*' value on the flag to Yes to indicate it is the end of the plot. Don't worry about the other fields for now.

Finally, in the object inspector, enter *Rescue the Princess* in the plot name field. The player will see this as the name of the quest in their journal. Save the plot.

Plot – Rescue the Princess

1. *Quest_Accepted*: Get Jack to help you rescue the princess.
2. *Talk_to_jack*: Find a way into the evil wizard's tower.
3. *Key_found*: Get the key from the bandit leader.
4. *wizard_Dead*: Kill the evil wizard.
5. *Princess_Rescued*: Return the princess to the king.

Creating an NPC

It's time to create an NPC that will send the player on the quest. We'll call him King Sam. Select [File] [New][Creature] and enter *King_Sam* as the resource name. We'll use the default values for most of the properties, but go down to the Name field and enter *King Sam*. Above the name field is the inventory field. Click on it and open the inventory window by clicking the browse button (...).

Let's give King Sam a nice set of clothes. Open up the _Global folder → Clothing → Noble → Human and choose some new clothes for our king. As you click on each set of clothes, you can see what it looks like in the preview window at the bottom. I'm going to use the *gen_im_cth_nob_b00* for this tutorial, but pick any outfit you like. You can add it to the NPC's inventory with the [Add] button or by double clicking on the item. Once it's added to the inventory list, click on the item on the right hand side and go to the slot field. Use to pull-down to select the chest slot, then click [OK]. This should show the king with his new clothes.

The last thing we'll do for our king is give him a different head. While it's possible to create our own head morphs, we'll just pick one from the list of original heads that came with Dragon Age. Go down to Head Morph field and open up the browse window. Pick a head you like and click [OK]. I'll use the *hm_arl100cr_gravedigger* head for King Sam.

Because King Sam is pivotal to our plot and we don't want him to die, set his Plot flag to true. He's not in any real danger, being safe in his castle, but it's a good idea to get in the habit of setting this for important NPCs. The only thing left to add to our king is the ability to interact with him and get our quest. We'll do that in the next section, Conversations. For now, make sure you save your changes to King Sam.

Conversations

As you can see from playing the game, conversations can get very involved with cut-scenes, voice over and animation. I'm not going to touch on those advanced subjects in this tutorial, but rather stick to a basic conversation.

Create a new conversation by going to [File][New][Conversation] and use *King_Sam_Conversation* for the resource name. Once you've created the conversation, you'll be looking at the root node at the top of the conversation. The bottom of the screen shows parameters associated with each line, spread across several tabs.

Enter all the text for our conversation. There are two ways you can add a line of dialog, *Insert Line* and *Insert Line After*. If you want to change speakers, you need to use the *Insert Line* option. If you want to add another choice for the current speaker, choose *Insert Line After*. You can do this with the buttons on the tool bar () or by right clicking on the line of text immediately preceding the line you want to add and use the drop-down menu. You can delete a line of conversation by selecting it and pressing the Delete key.

Enter these lines of text into the dialog editor. To enter text, select the proceeding line (root for the first line) and then hit *Insert Line*. The text is entered at the bottom, under the dialog tab. Feel free to copy/paste each line.

- Thank you for coming so quickly.
  - <act>Ignore the king.</act>
  - What can I do for you, Sire?
    - My daughter has been kidnapped by an evil wizard and I need you to rescue her.
      - I shall return with your daughter soon.
        - Thank you. Talk to Jack before you leave. He will aid you on your quest.
          - I will, Sire.
          - I don't need any help.
      - What's in it for me?
        - If you bring her back safely, I offer you her hand in marriage.
          - I've met your daughter and I must respectfully decline.
            - <desc>sigh</desc> Then I will grant you a barony.
- Any news of my daughter?
  - None yet, Sire.
  - I have rescued your daughter.

Now we need to attach this conversation to the king. Save your work by clicking the save button and then go over to the palette window. Make sure the creature button () at the top of the palette is selected. You should see King_Sam on the creature list. Double click his name to edit the creature. Go to his *Conversation* property and change it to *King_Sam_Conversation*. Save the changes and go back to the area (either click on the area tab or going to the palette window and click on the area icon (), then select castle from the area list).

In the palette window, with the creatures button selected, click on King_Sam and then click in the view area to place him. Put him at the top of the steps, and rotate him to face the player's starting point.

Our king is finally ready to test. Save the area and, with the area still open, go to [Tools][Export][Export with Dependent Resources]. Open the module in the game and try talking to the king. If you get stuck in the conversation and are unable to continue, you can press escape to exit out. This occurs on conversation lines that don't end with [end dialog].

As you talk to him, notice the tags we put in. The first one is the action tag <act>some text</act>, indicating that an action is preformed rather then a line spoken. The text between the two flags appears in parentheses and is italicized automatically. You can combine text and actions together. For example, "You have to catch me first! <act>run away</act>" would appear as "You have to catch me first! (*run away*)." Be aware that using the <act> tag doesn't actually cause an action to happen, that normally requires scripting.

The second tag is the description tag <desc> and </desc>. Like action tags, description text is also put in parentheses and italicized. Description tags are used for several purposes:

- Unspoken sounds, such as a sigh, laugh or cough. Example: "I'm feeling sick <desc>cough</desc> and can't come in to work today."
- Character abilities, such as charm or coercion. Example: "<desc>Charm</desc>You look especially beautiful today."
- Description of items. Example: "<desc>The small black book contains a list of names and addresses</desc>"

Let's return to the editor and make the conversation actually do something. In the tool set, go to the conversation (click on the tab if it's there, or open it up from the palette after clicking on the conversation icon,).

Start by having the conversation give our quest to the player. Select the line, "*I shall return with your daughter soon.*" At the bottom, click on the plots and scripting tab. In the action section, click the browse button (...) to find the plot we want to use. Select the *rescue_princess_plot* and hit [OK]. On the flag field right below our quest, choose *Quest_Accepted* from the pull-down. By default, the plot flag will be *set,* making the quest active and automatically adding it to the player's journal. Notice that there is a blue 'A' in front of the line indicating that an action has been assigned to this conversation line.

If you were to run the module right now, the player would get the quest, but when they talked to the King a second time, he would go through his whole spiel and try to give the player the quest again. While this wouldn't break anything, it's not what we want. Select the first line, "*Thank you for coming...*" Go to the plot field under the condition section of the plots and scripting tab. Since we've already added our plot to this conversation, it will now appear on the pull down. Select it and, under the Flag field, chose the *Quest_Accepted* flag. Change the field to the right of the plot to '*is false*'. This will only show this line of the conversation if this flag has not been set (meaning the player hasn't accepted

the quest). A blue C at the beginning of this line will mark that there is a condition set for it.

We also want to set a condition for our last line, “I have rescued your daughter.” Set the condition for the *Princess_Rescued* to *'Is True'* so it only shows up after we've rescued her.

Lets also let the player of accept either of the two offered rewards, either the princess's hand in marriage or the barony. Since we already have a line which accepts the quest, we'll link the same line to the responses under the offers. Select the “I shall return with your daughter” line and hit ctrl-c to copy it. Now, RIGHT click on the “*If you bring her back safely...*” line and choose the *Paste as Link* option from the drop-down menu. You will see a grayed out line that is a duplicate of the one we copied. This means that when the conversation reaches this point in the game, it will jump to the line we linked and continue from there.

Paste the same line as a link under the line offering the barony. Save this conversation, and then, with the conversation still open, go to [Tools][Export][Export with Dependencies]. Once it's finished, test your conversation in game. Make sure you can only get the quest once and that it's assigned correctly. If you talk to the king after you have received the quest, he should have a different response for you. You can also open your journal to see what the next step of the quest involves.

Adding a New Party Member

Now, we'll create another character that the player can have join their party. Start by creating a new conversation and call it Jack_Conversation. Add the following lines from the gray box to it.

- The King wants to speak with you.
  - I'll go find him.
- Hello there.
  - Join my party.
  - Never mind.

In this conversation the NPC, Jack, has only two speaking lines. Since they are on the same level, the game will pick the first option and completely ignore the second, which is what we want to happen until after the player has spoken with the king. Once the player has the quest, they can then add Jack to their party.

Set a condition on the first line using the *rescue_princess_plot* created earlier. This time, check the *Quest_Accepted* flag and make sure the condition is set to *is false*. If the player doesn't have the flag set, this line will be shown, otherwise it will drop down and let the player ask Jack to join them.

To add Jack to the player's party, use a special plot provided by Bioware for party functions. Select the line, “Join my party.” and go to the action section under the plots and scripting tab. Open the global folder, and then the party folder and find the gen00pt_party plot. With that plot selected, pull down the flags drop-down and find GEN_HIRE_FOLLOWER.

Talking to Jack is one of the steps of our quest. To update the plot, select the “Hello there.” dialog line and add the *rescue_princess_plot* to the action field, then set the *Talk_to_Jack* flag.

Save the conversation and you now have a simple conversation that will add a follower to the group.

Create Jack just like you created the king ([file][new][creature] and use jack as the resource name). Change his name property to Jack, set his conversation file to Jack_Conversation, and set his class to Warrior. Scroll down to head-morph and find a head you like. To make Jack participate in combat, you need to give him some basic AI. Go down to packages, and change General Package Type to *Party Members*. Set Package to Generic – Melee, and Package AI to AIP_Follower_Generic_Melee.

Now, scroll back up to inventory to give him some basic gear. In the inventory window, go to _Global and give him the following equipment:

- Armor, Boots → gen_arm_bot_lgt_ltr (Leather boots)
- Armor, Chest → gen_im_arm_cht_lgt_rlr (Studded Leather Armor)
- Armor, Gloves → gen_im_arm_glv_rlr (Studded Leather Gloves)
- Armor, Shield → gen_im_glv_sml_wdn (Small Shield)
-
- Miscellaneous, Quick Items, Health Potions → gen_im_qck_health_101 (Lesser Health Poultice)
-
- Weapons – Melee, Longswords → gen_im_wep_mel_lsw_lsw (Long Sword)
- Weapons – Daggers → gen_im_mel_dag_dag (dagger)
- Weapons – Daggers → gen_im_mel_dag_dag (dagger)

Set the Lesser health poultice stack size to 2.

Save Jack and go back to the area editor. Select creatures on the palette and place Jack near the exit to the area where the player will have to run past him to exit the room.

Area Transitions (Doors)

To add a transition out of this area, we need to make a new area for the player to transition to. Create a new area with a resource name of *forest*. Set the name field to *Bandit Forest* in the object inspector, then open the browser for the A*rea Layout* and find the level BRC101d.

We'll set the entrance for the area near the arch in the northwest corner, marked 'A' on the map. Put a way point there ([Edit][Insert Waypoint]). Be sure to rotate it so the player is facing down the path when they transition so they don't end up running right back out of the level. Set both the way point's name and tag to *from_castle* in the object inspector.

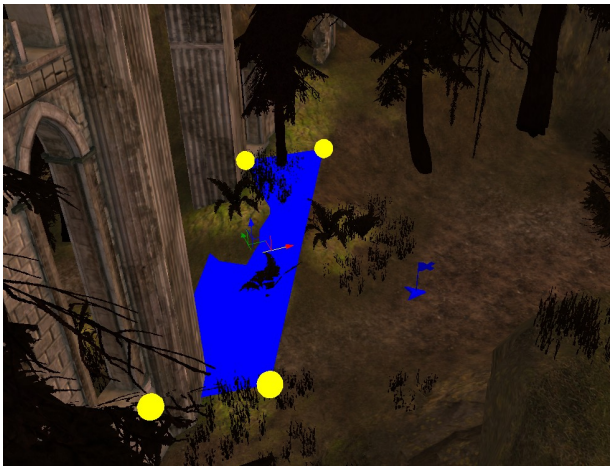
Set the way point's map note by changing *MapNote enabled* to True and then enter “To the Castle” as the note. Also, change the MapNote type to *Area Exit*. This changes to icon on the map to an area transition icon.

Save this area and go back to the Castle area to add the transition from the castle. At the main entrance to the castle, we will place another set of doors. This time, we need to use an area transition door rather then a normal door. Under Placeables on the palette, go to Global, Area Transitions, select genip_at_double (door) and place it over the entry to the hall. The area transition information is set in the Variables field. Use it's browse button to show the list of variables available for the door. The two we care about are PLC_AT_DEST_AREA_TAG and PLC_AT_DEST_TAG. Change the value of PLC_AT_DEST_AREA_TAG to *forest*. This is the resource name for the area we want to transition to. Change the PLC_AT_DEST_TAG to *from_forest*, the tag of the way point the player will enter the new area at.

We also need to add a transition leading back to the castle. While still working in the castle area, place another way point in front of the doors and change it's name and tag to 'from_forest'. Make it a map note and change the MapNote field to 'Exit'.

Area Transitions (Triggers)

Back in the forest area, we need to add the transition back to the caslte. This time, we'll use a trigger for the transition. A trigger defines a space that triggers an event when it's entered, such as springing a trap, starting a cut-scene, or running a script. In our case, it'll cause the player to transition back to the castle.



Start with a new trigger ([File][New][Trigger]) and give it a resource name of *Castle_Transition*. You'll be presented with the properties of the trigger. Go to the Variables field and hit the browse button. Similar to the way point transition, find the TRIGGER_AT_DEST_AREA_TAG and set its value to the tag of the area to transition to, c*astle*. Set the TRIGGER_AT_DEST_TAG to the way point tag, *from_forest*. Save the trigger. Note that tags are case sensitive.

Once the trigger is created, select it from the palette. The mouse cursor will turn into a pair of cross-hairs. Triggers can be any shape you want, with multiple corners, created by clicking their locations in the area. Lay down the trigger by clicking four points in front of the arch making a rectangle completely blocking the path. Double click on the last one to close off the area. The trigger doesn't need to be directly on the ground and will probably pass through the ground at some points. As long as the player runs under, over or through it, it will be triggered.

You can adjust the size of the trigger by moving the yellow balls, or move the entire trigger by selecting the blue area. Make sure that the way point for the transition from the castle is in front of the trigger and not on top of it. You can test the area to make sure the transitions are working in both directions. Do an export of both the castle and the forest areas before you try ([Tools][Export][Export with Dependent Resources]).

Generating a Character

So far, the module has been using a default level 0 character with 1 health point. To force the module to go through the character generating process, we need add a little script to the module. Make a new script ([File][New][Script] with a resource of gen_char. Copy the script in the box below and paste it into script window.

```
#include "events_h"
#include "global_objects_h"

void main()
{
    event ev = GetCurrentEvent();
    int nEventType = GetEventType(ev); //extract event type from current event
    int nEventHandled = FALSE; //keep track of whether the event has been handled
    switch(nEventType)
    {
         case EVENT_TYPE_MODULE_START:
         {
            PreloadCharGen(); //preloads resources for character generation
            StartCharGen(GetHero(),0); //initiates character generation
            break;
         }
    }
    if (!nEventHandled) //If this event wasn't handled by this script, let the
core script try
    {
        HandleEvent(ev, RESOURCE_SCRIPT_MODULE_CORE);
    }
}
```

Compile the script using the compile button on the toolbar and save it. Once finished, open the Module Properties dialog ([Open][Manage Modules]). In the scripts field, hit the browse button and find the gen_char script, or just type it in the field. Click [OK] to save the the changes, and export the main area. The next time the module is launched, the player will be presented with the character generation screen.

Adding Hostile Creatures

Adding basic hostile creatures to the game is straight forward. We're going to place some wolves as the first encounter. With the Bandit Forest area open, select the creatures () button on the palette. Find the wolves under natural beast.

In the area view, move to the area marked 'B' from the map on Page 10, right around the corner behind the tree. With a wolf selected in the object palette, place three wolves on the road and you have a simple encounter.