G54GAM – Lab Session 4

Game Description

Tanks

This lab session is based on creating a tank combat game for two players. Each player steers a tank through the arena with the aim of destroying the other player's tank using their weapon. Destroying an enemy tank is rewarded with a point. The players share a window that provides two player-specific views onto the arena.

Mechanics

The game takes place in a large arena, or level, with walls providing features to hide behind and steer around. Each player steers their tank through this arena, searching for the other player. Each player has a weapon that fires bullets which damage their opponent. Each tank has a different weapon. Player one fires a bullet that moves quickly and bounces off walls. Player two fires a bullet that moves slowly but which seeks out and homes towards the enemy. Collision with a bullet fired by the enemy reduces the player's health, and when the player's health reaches zero the enemy scores a point.

Controls

Each player controls their tank with a different set of keys. Player one uses A and D to steer left and right and W and S to move forwards and backwards. The space bar fires their weapon. Player two uses the arrow keys to steer and move, and the Enter key fires their weapon.

Create a Steerable Tank and the Arena

The first task is to create an arena in which our game takes place and two tanks for the players to control. As always, we need to begin by creating resources.

- **Sprites**
  - A **sprite** for the wall that we will use to define the edges of the arena and some interesting features to navigate around and hide behind
  - A **sprite** for each tank, with 60 frames for the different directions that the player can steer the tank in
- **Background**
  - A **background** tile to make the arena look more interesting

Creating the tank sprites would take a long time by hand, so you can download two different tank sprites here (right click and "save as" tank1.gif and tank2.gif):

http://www.mrl.nott.ac.uk/~mdf/teaching/G54GAM_LAB04_Resources/

Create the wall sprite in the usual way, but use the **load sprite** function to import the ready made tank sprites. Create or find a background tile of your choice.

Next create the arena:

- Create a wall **object** using the sprite that you have just made, remembering to make it **solid**.
- Create a new **room**, making it quite large (at least 1000 x 1000 pixels), defining the edges of the room and some features by placing instances of the wall object in it.
- Assign the room the **background** that you have just made.

Now we need to make the tank. We'll make one first, then we can duplicate the object to make a second. We'll make two different objects so we can easily tell the difference between them and handle the different events accordingly.

- Create a tank **object** and assign its **sprite** to be one of the tank .gifs you downloaded.

There are four kinds of events we need to handle to make the tank move and be rendered in the right way

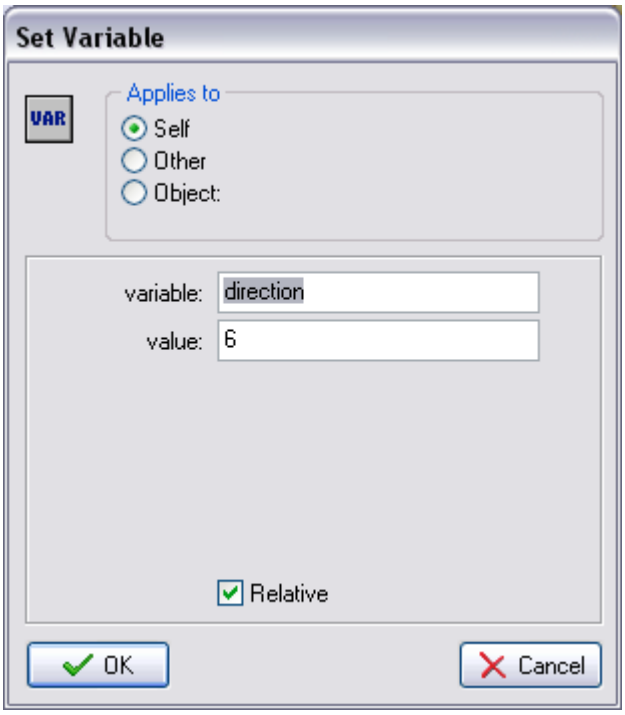
- **Create** event – set the friction of the object
- **Keyboard** event – changes its speed and direction
- **Step** event – *set* the correct frame of the sprite based on its direction
- **Draw** event – *draw* the right frame and not just play the .gif frames in order

Let's look at each one in turn.

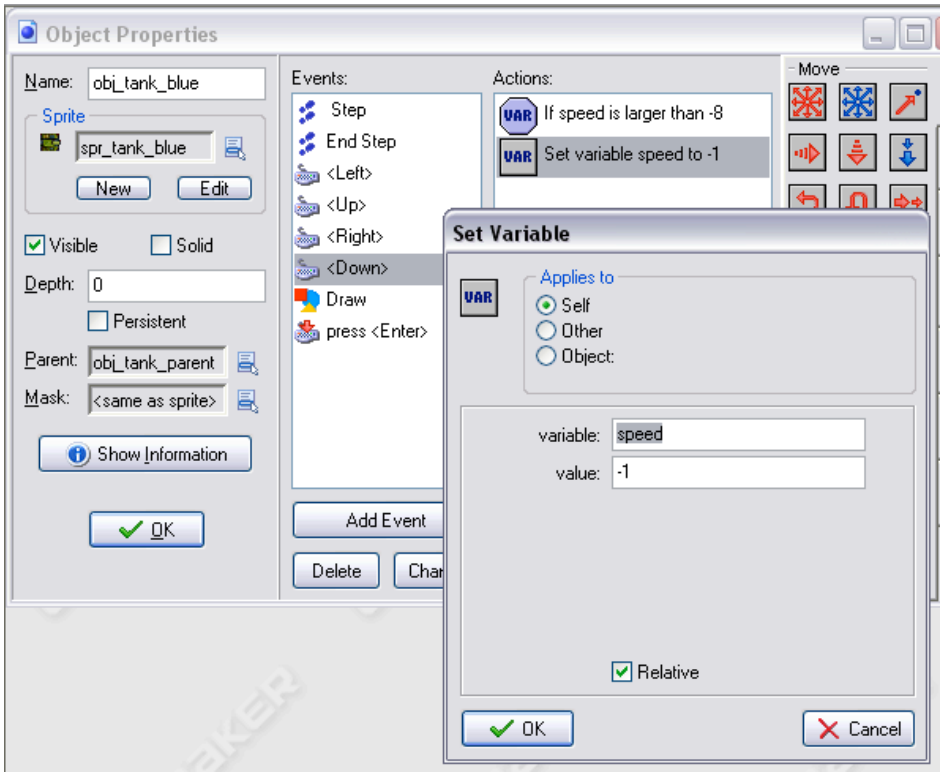
- **Create** event. Add a **Set Friction** action, setting the friction to **0.5**. This means that every step, 0.5 pixels per step will be subtracted from the speed of the tank, making it slow down quite quickly

- **Keyboard** events. There are two things we need to do here, change speed and change direction.

    - Add a **Keyboard** event for the left arrow key. When this is pressed, we want the tank to turn anti-clockwise – or, in Game Maker coordinates, increase its **direction** variable. So, for this event add an action that sets the **direction** variable to be 6 and relative. Remember, we have 60 frames in our sprite so we have an increment of 6 degrees (360/6 = 60)



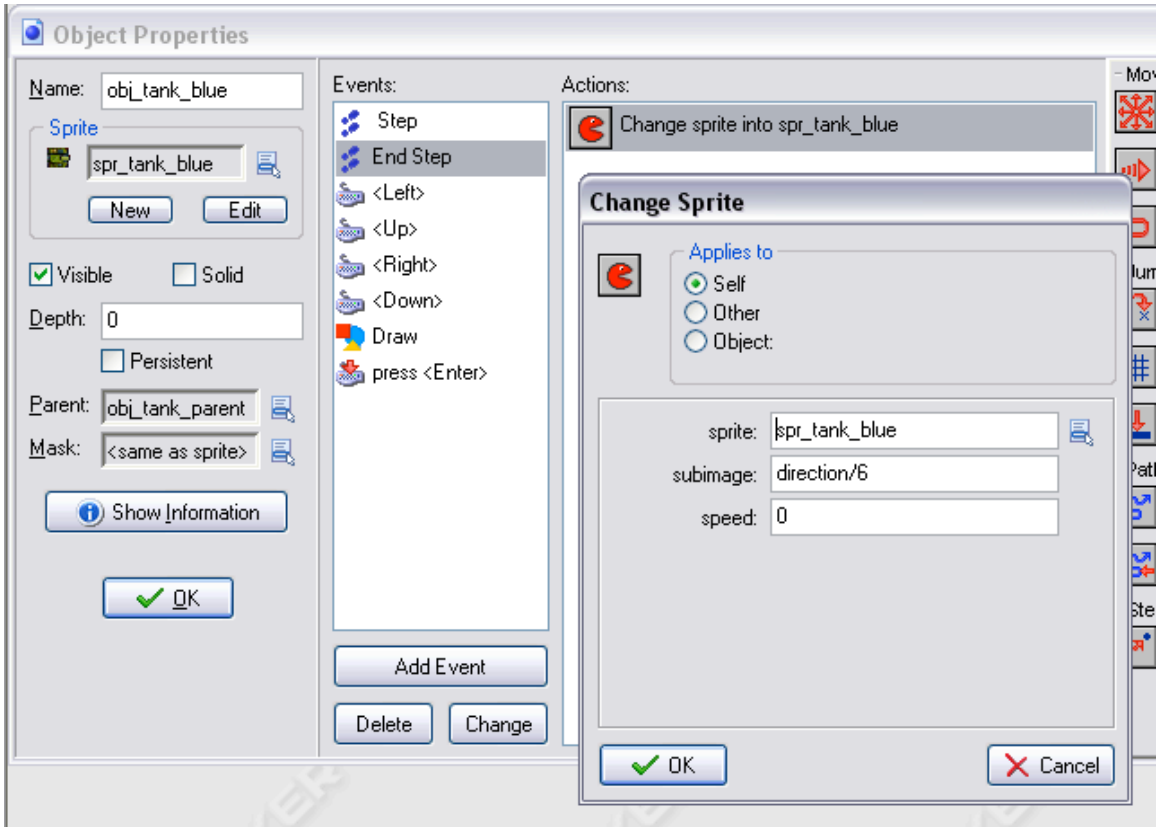
- Add a similar keyboard event for the right arrow key (subtracting 6 this time) so we can turn clockwise



- To move forwards and backwards we need to modify the speed of the tank. Add a keyboard event for the up arrow, and to this add an action that adds 1 to the instance variable **speed**, remembering to limit it at a reasonable value with an **if** statement so the tank doesn't move too fast
- Add a similar keyboard event and action for the down arrow key, remembering to add -1 to the speed to make the tank move backwards.

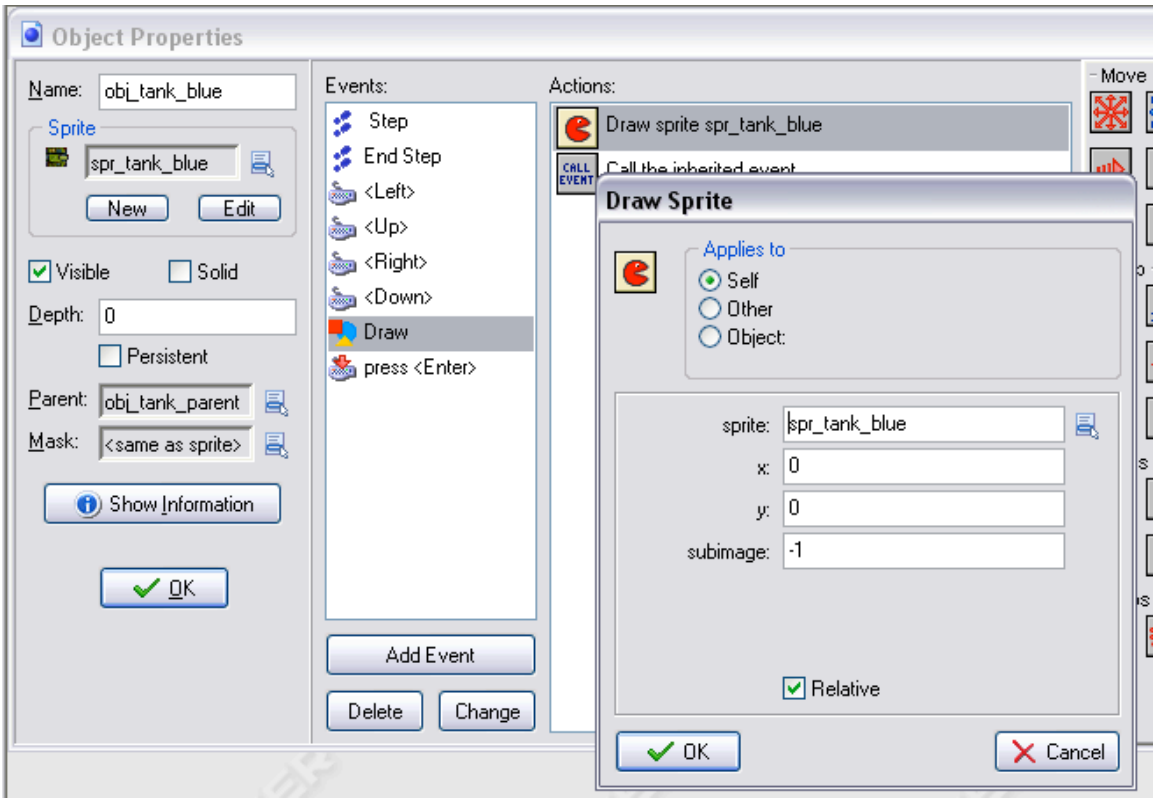
Add a tank object to the room and test the game – the tank is probably spinning and looks odd, but you should be able to drive it around using the arrow keys. If that works as planned, lets now fix how it looks

- **Step** event – every time the model updates, we want to work out which frame of the sprite to draw so the tank appears to be pointing in the right direction. This time we'll handle an **End Step** event so it occurs after Game Maker has finished moving all of the objects. Add a **Change Sprite** action, but set the **subimage** field to be **direction/6**. Think about what this means – we have 60 frames in our sprite, and direction is an angle between 0 and 360 degrees. So **direction/6** will calculate which frame between 0 and 60 that we should show for the tank's current rotation, where frame 0 shows direction 0 degrees, frame 30 shows direction 180 degrees and frame 59 shows direction 354 degrees.



- **Draw** event. The last thing we need to do is to stop Game Maker making the tank spin – its trying to be helpful by automatically animating the sprite, but in

this case we don't want that to happen. Handle the **Draw** event for the sprite, and add a **Draw Sprite** action, setting the sprite to your tank sprite, and the **subimage** to **-1** as show below. -1 tells Game Maker not to change the sprite when it draws it, because we've already change it in our **End Step** event above.



Finally handle how the tank interacts with its surroundings, i.e. what happens when we drive the tank into the wall

- Handle the **collision** event between the tank and the wall. You can make the tank bounce away from the wall by adding an action to set the **speed** variable to be **–speed** – which will make it bounce backwards when it hits something

Test the game – we should now have a tank that we can drive around the arena using the arrow keys. If it doesn't work something's gone wrong, so review the previous steps and try and work out what you've missed.

Multiplayer Game

Creating a multiplayer game should be easy now that we've create a tank and the arena. First we need to create a second tank for the second player to use

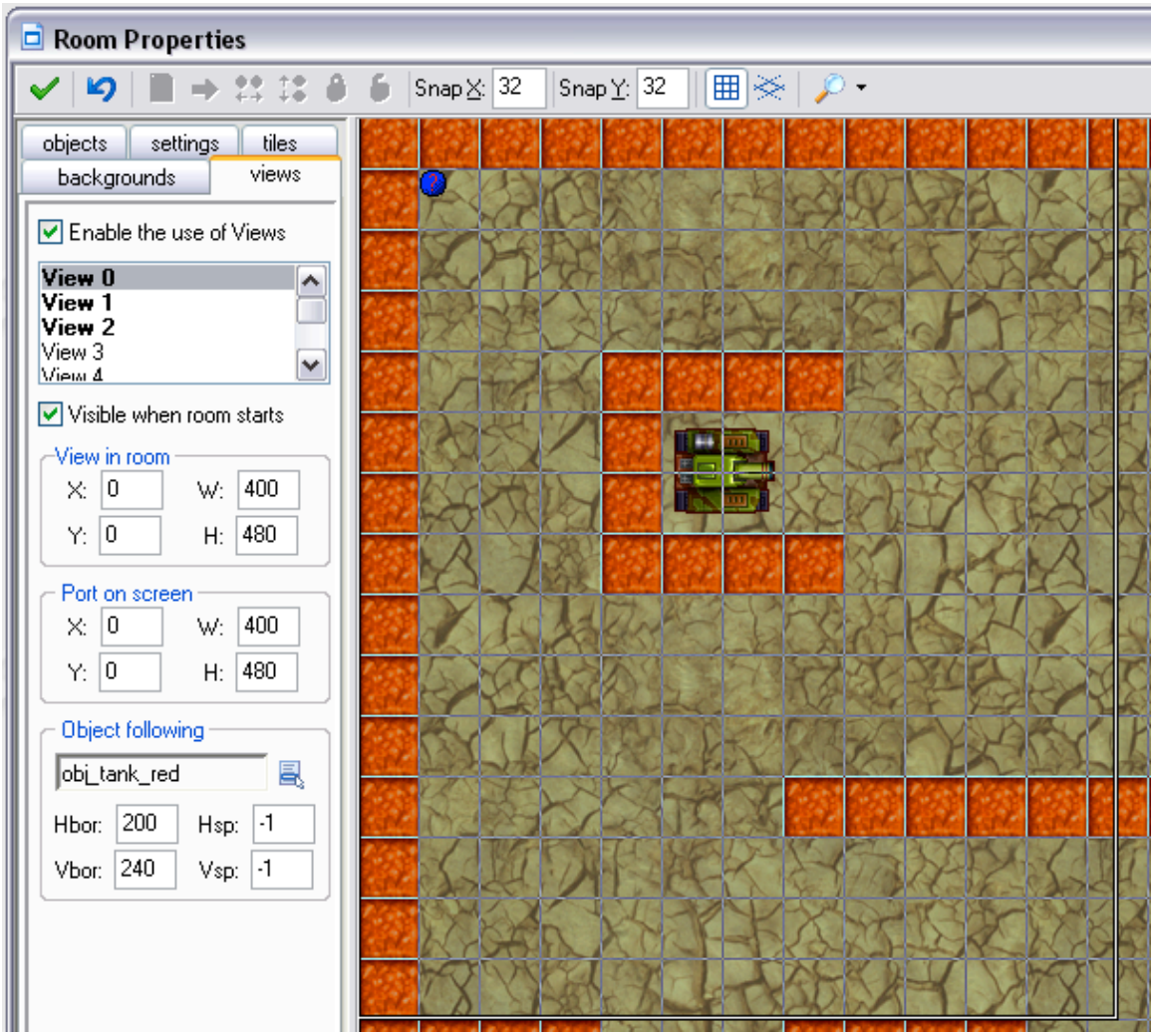
- **Duplicate** the tank object giving it a sensible name (for example obj_tank_blue), assign it the second tank sprite, and change its **Keyboard**

events to use the W,A,S and D keys to move (this is the classic key arrangement for a first person shooter).

- Add an instance of the new tank object to the room

Test the and check that you have two tanks that you can drive independently.

Next, add multiple **views** to the room, using the views tab for the room that we made earlier, adding a view for each player and a central view that gives a map overview of the arena



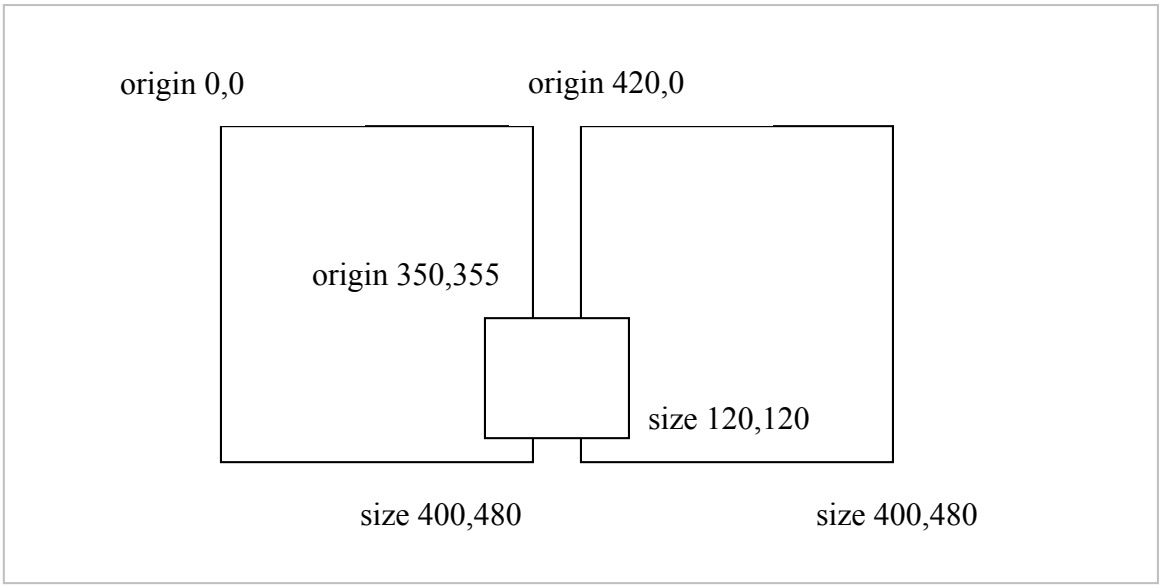
A **view** has a number of properties that we can change:

- The size of the view with respect to the room – which bit of the arena we're looking at
- The size of the view with respect to the window on the screen (the **port**) – the layout of the different views on screen
- The object that the view tracks, if any – the two tanks

- The border of the view that will force the view to update and move when the tracked object moves outside it

Create the three views for the players, thinking about the different coordinate systems in play:

- The size of each view on the screen and its layout
- What object each tracks with relation to the size of the room



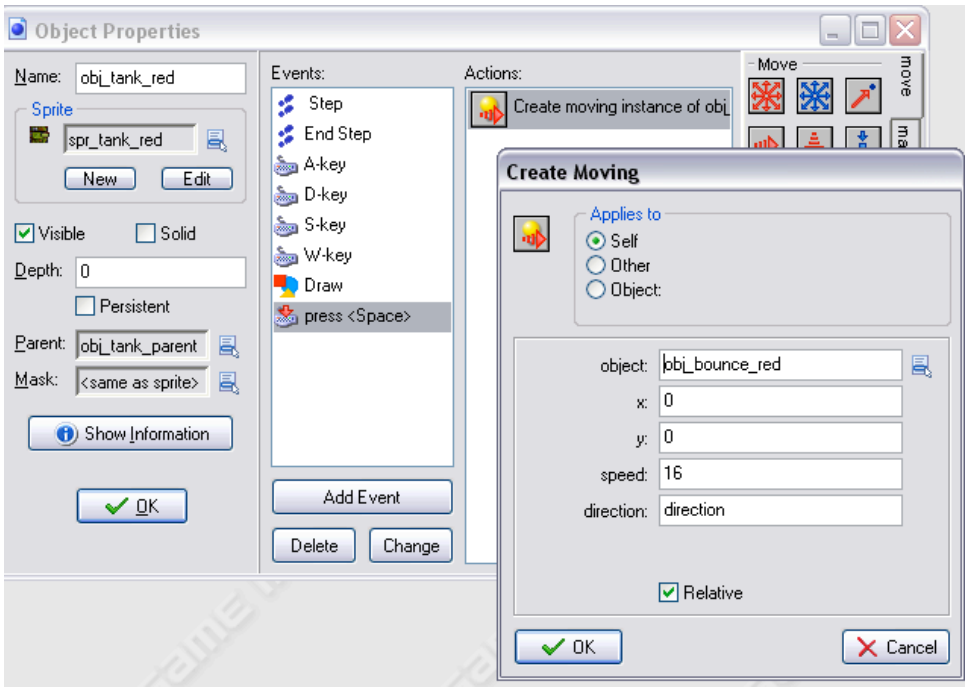
Test the game and make sure that the views appear and behave as expected

- Is the border of each view sufficient that each player can see where they are going?

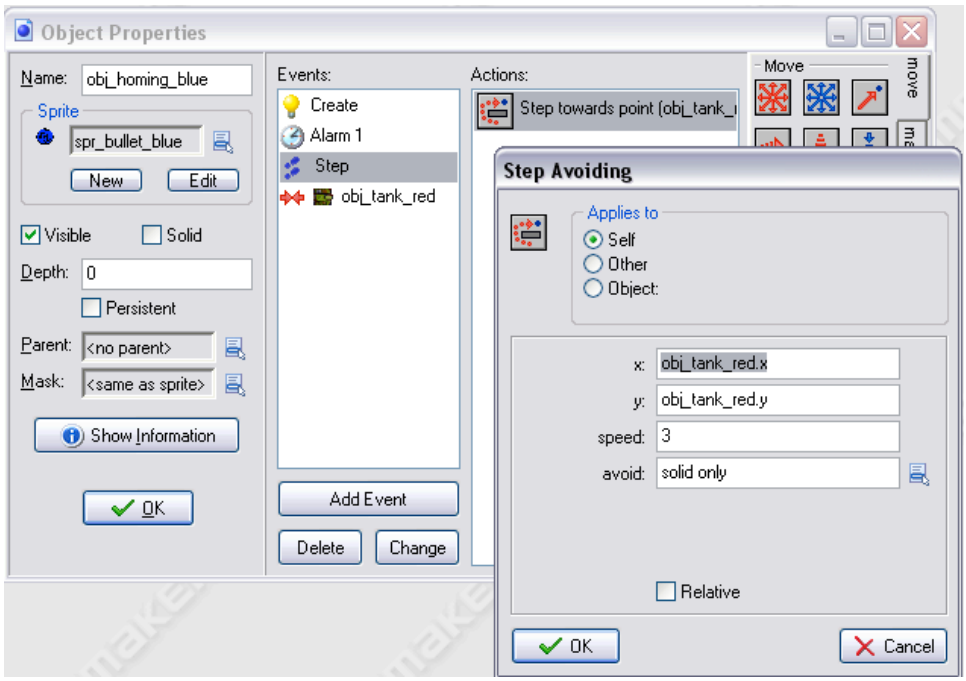
Game Play

By now we should have an interesting environment for our two players to explore. The final task is to turn this into an interesting game, and in this respect we can reuse many features from the shoot-em-up from previous sessions.

- Create bullets that are fired from each tank when the appropriate key is pressed
    - Use the **create moving** action to create an instance of a bullet object that moves in the same direction as the tank is facing



- Create different bullet objects depending on which tank fired a weapon. A bouncing bullet just needs to handle the **collision** event using the **bounce** action to bounce against walls. A homing bullet needs to handle the **step** event using the **step avoiding** action to move towards the enemy player.



- Handle the **collision** between bullets and tanks to decrement a local damage variable, and calculate whether a player has lost a point or a life.