CRASH ZONE

SECTION 1

Crash Zone is the next Construct game that you'll build. It's a demolition derby game that features two player controls.

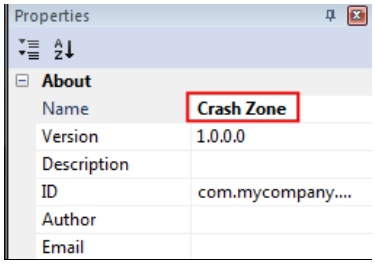
The main feature of the game is varying damage based on where the car and collision occur. Other features will include different surfaces that will cause the cars to handle differently, modifying damage based on speed, and showing damage on cars through different images.

SECTION OBJECTIVES

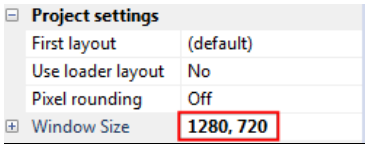
Before you start, read the section objectives below. These objectives will be completed at the end of this section.

- Create two cars that will move using different keyboard controls
- Enabling the cars to collide and remove health
- Add a multiplier that relates the damage dealt to a car's speed

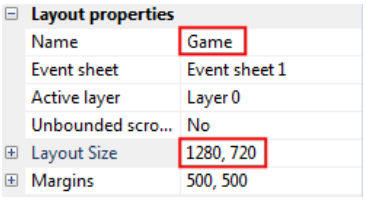
To start, open a new empty project, in the About section of the Properties rename your game to **Crash Zone**, and save your game.



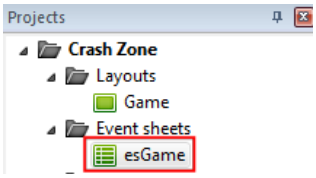
Next set the window size for your game. In the Project settings section for your game, change the Window Size to **1280, 720**.



Select Layout 1 from the Projects bar. In the Layout properties section rename the layout to **Game** and change the Layout Size to **1280, 720**.

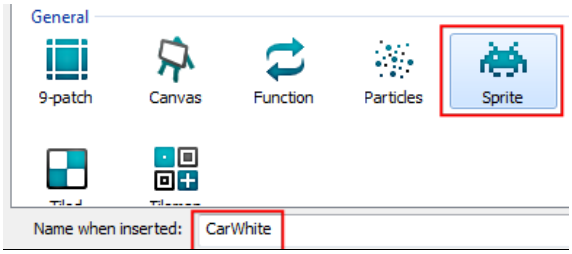


One last thing before you start building your game. In the Projects bar, right-click on Event Sheet 1 and click rename. Rename your event sheet to **esGame**.



OBJECTIVE 1 - CREATING CARS

You're now ready to start on the first objective, creating two cars that are controlled using different keys. To begin, go to your Game layout and insert a new object. This object will be a **Sprite** named **CarWhite**.



Click anywhere on your layout to insert your Sprite and bring up the image editor.

In your image editor, Resize your sprite to a Width of **120** and Height of **60**.

Select the fill tool and in the Color Palette select any color you wish. Click on the sprite to color your image and close the image editor

Now that you have your sprite inserted you're going to add a new Behavior to the sprite. With your CarWhite sprite selected click the **Behaviors** link it the Properties bar.

In the newly opened Behaviors window click the **Plus** button to add a new behavior. In the Add behavior window select **Car** from the Movements section. Click **Add** to insert the behavior.

The Car behavior is a built-in Construct behavior that will allow you to use the arrow keys to move the sprite and simulate the movements of a car.

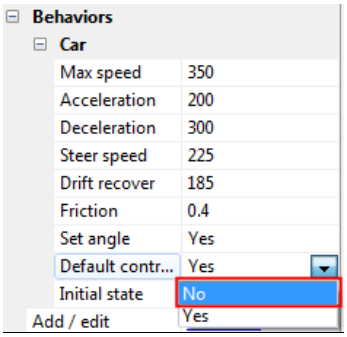
Run the layout and use the arrow keys to test this behavior.

Add the second car to your game

- Create a new Sprite named CarBlack
- Change the sprite image to an all black box that is 120 width and 60 height
- Add the Car behavior to the object

Now that you have your object inserted and the car behavior added you're going to change a setting in the Car behavior properties.

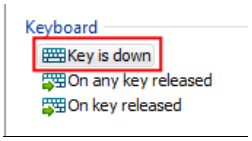
Next to Default controls click **Yes** to bring up the dropdown list. In the list select **No**.



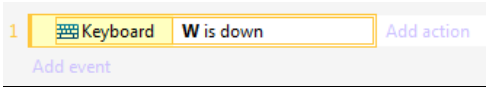
Turning off the default controls will allow you to change which keys are used to move the CarBlack object. You'll do this by adding events to your event sheet. First you'll have to add the Keyboard object to your project.

Insert a new object and select **Keyboard**.

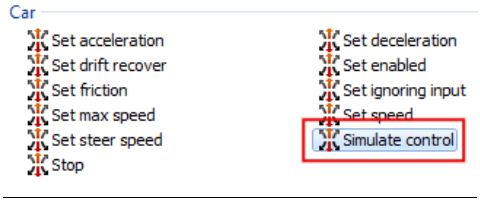
With the Keyboard object inserted, go to the esGame event sheet and add a new event. Select the Keyboard object and for the condition select **Key is down**.



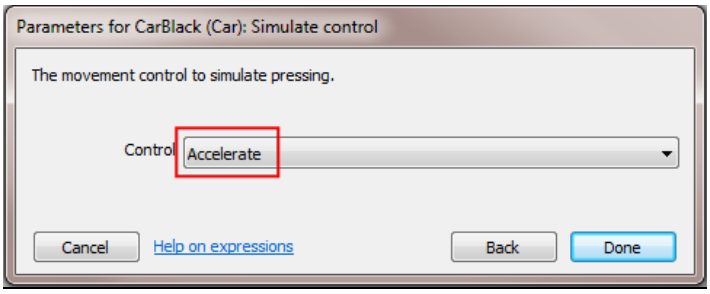
Press the **click to choose** button and when the choose a key window appear press the **W** key, then click **OK** to close the window, and **Done** to insert the event.



Once your event is added, click the **Add action** link and select the **CarBlack** object. For the action select **Simulate control** under the Car section.



In the parameters window, select **Accelerate** and click **Done** to insert the action.

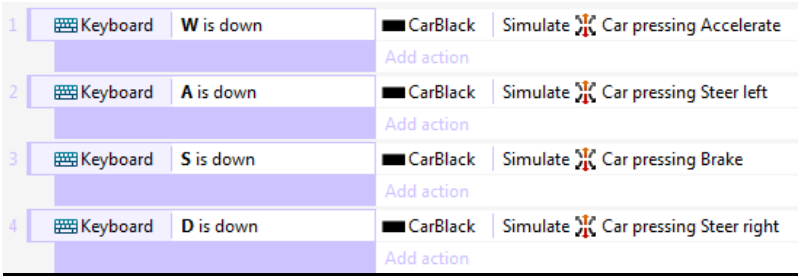


What this event will do is allow the acceleration of the black car to be controlled by the W key instead up the up arrow. This allows you to be able to control both cars on the same keyboard.

Do the following three task to complete the controls for the black car

- Create an event that has the A key steer the black car left
- Create an event that has the S key brake the black car
- Create an event that has the D key steer the black car right

IT SHOULD LOOK LIKE THIS WHEN YOU'RE DONE



Run the layout to test your controls. If done correctly the CarWhite object should be controlled by the arrow keys while the CarBlack object is controlled with W,A,S,D.

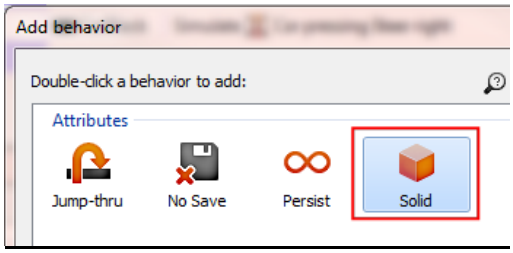
STOP! READ THIS BEFORE CONTINUING!

Always buckle up and save your games frequently.

OBJECTIVE 2 - COLLISIONS AND HEALTH

You may have noticed when you ran your layout that your cars currently don't collide. We can solve this issue by simply adding a new behavior to each car object.

Select one of your car objects and in its Properties Bar select the **Behaviors** link. In the Add behavior window, select **Solid** from the Attributes section.



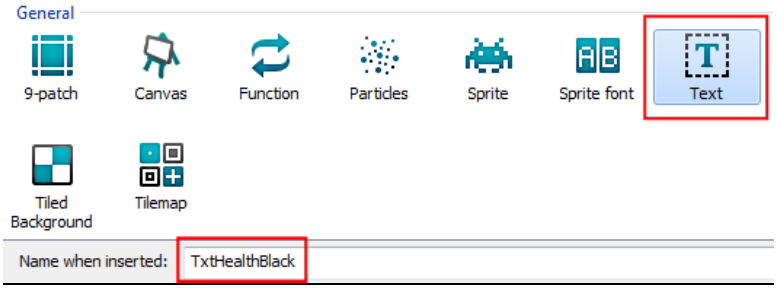
Repeat this for the other car object so both cars have the Solid behavior on them.

Once both cars have the behavior you can run the layout and see that the cars now collide instead of passing through one another.

Now that you have the cars colliding, you'll want to add a variable for health to each car. For each car object add an Instance variable named **Health** and set the Initial value to **500**.

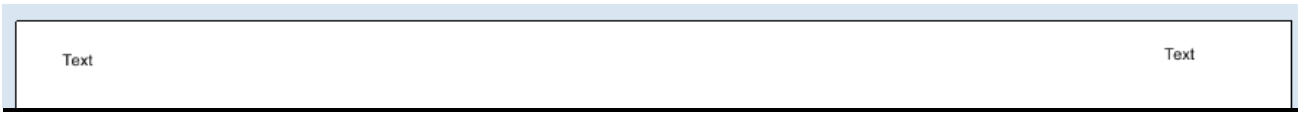
Next you'll add a couple objects to display the health of each car. You're going to use just a simple text object for this, but in the final version of the game you'll make a health bar.

Make sure you're on your Game layout and insert a new object. For the object type select **Text** and name your object **TxtHealthBlack**. Click **Insert** and using your crosshairs insert your new text object in the top-left of your layout.

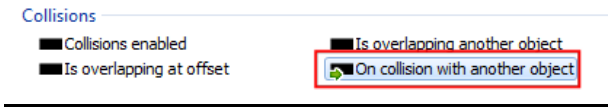


Repeat this to create another Text object named TxtHealthWhite and insert this object in the top-right of the layout.

IT SHOULD LOOK LIKE THIS WHEN YOU'RE DONE

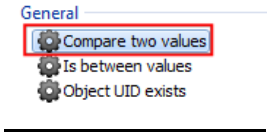


Now that you have all your objects inserted you're ready to create the events for the collisions. Go to your event sheet and add a new event. For your object select **CarBlack** and for your condition select **On collision with another object**. In your Parameters window, select **CarWhite** as the object and click **Done** to insert your new event.



Before you add an action you'll need to add a sub-event to your new event. In this sub-event you're going to have a condition to test the speed of the CarWhite object. You'll need a condition to prevent the cars from losing health when they "rub" against one another after a collision.

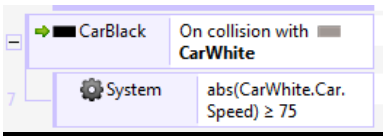
On your newly created event, right-click and add a new sub-event. You're going to to use the **System** object and for the condition select **Compare two values** under the General section.



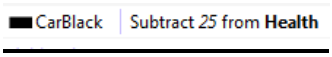
For the first value in your comparison you're going to select the speed of CarWhite. Since your cars will be moving both forward and backwards during gameplay you'll need to turn all the speed values into a positive number. To do this you'll use the expression **abs**. This will get the absolute, or positive, value of a number.

In the First value field, insert the absolute value expression by typing **abs(**. Inside the parentheses you'll access the speed of the CarWhite object inside the Car behavior. To do this type **CarWhite.Car.Speed**. Insert your closing parenthesis to complete your expression, **)**.

To complete our parameters change the Comparison to **Greater or equal** and change the Second value to **75**. Click **Done** to insert of complete condition which will test that speed of CarWhite is greater or equal to 75 in order to register damage.



With conditions set add a new action onto your sub-event. Select **CarBlack** as your object and for the action select **Subtract from** under the Instance variables section. Now in the Parameters window change the Value to **25** and click **Done** to insert your action.



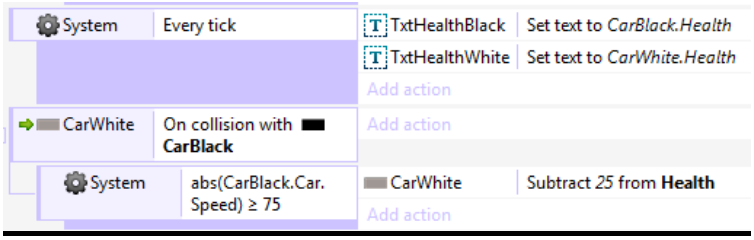
Next you'll want to set up an event to set the text of the TxtHealthBlack object to the actual health variable. To do this, you'll use the every tick event. Add a new event to the event sheet using the **System** object and the **Every tick** condition.

Now add an action to the every tick event. Use the **TxtHealthBlack** object and select the **Set text** event. For the Parameters set the Text field to **CarBlack.Health**.

This event makes sure our TxtHealthBlack object is accurate to our current health. Next you'll set up the collision events for the CarWhite object

- Add a new event for CarWhite on collision with CarBlack
- Add a sub-event to test the absolute value of CarBlack is greater or equal to 75
- Add an action to the sub-event to subtract 25 from CarWhite's Health
- Add an action to the every tick to set the text of TxtHealthWhite to CarWhite's Health

IT SHOULD LOOK LIKE THIS WHEN YOU'RE DONE



Run your layout and perform a few tests to make sure your events are correct. Take turns keeping one car idle and hitting that car with the other. The idle car should lose health while the moving car doesn't. If both cars are moving at over 75 pixel/sec on a collision both cars should lose health.

If everything appears to be working you're ready to add your damage multipliers.

SAVE YOUR GAME NOW!

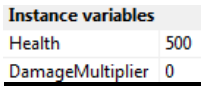
Like a hockey goalie why don't you make a save.

OBJECTIVE 3 - ADDING DAMAGE MULTIPLIER

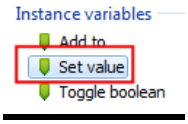
Your damage multiplier will be less of a multiplier than a divider. In order to get the maximum value from a collision a car will have to be traveling at its maximum speed. Anything less we will multiply the damage by what percentage of the maximum speed the car is traveling.

To begin, you'll first need to set up some variables within your game.

The first variable you'll add will be the damage multiplier, which you'll set as an instance variable. From the Objects Bar select the CarBlack object. In its Properties Bar, click the **Instance variables** link. In the Instance variables window add a new variable named **DamageMultiplier**. You can keep the initial value at zero since you'll dynamically change the value using events.

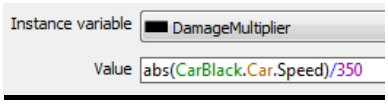


While you have CarBlack selected take note of the value of Max speed (350) in the Car behavior, you'll be using this value soon. Select your event sheet and on the Every tick event **Add an action**. Select **CarBlack** as the object and in the Instance variables section select **Set value** as the action.



Select **DamageMultiplier** as the variable you're setting. For the value you'll need to divide the current speed of the car by the maximum speed. This will return a decimal value you can use in your collision event. If you remember earlier we need to use the absolute value of the speed, we will do the same here.

In the Value field type **abs(CarBlack.Car.Speed)/350**.

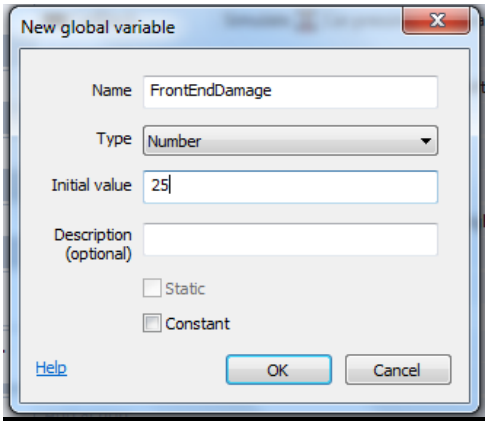


- Add an instance variable named DamageMultiplier to the CarWhite object
- Add an action to the every tick event to calculate the CarWhite DamageMultiplier

Now that your damage multipliers are set you're going to add a global variable for the damage. In the next section, you'll expand the events in the car collisions by adding events that calculate where collisions happen. In these events different damage values will be giving based on where the cars collide, so to keep things simple you'll add variables for these different damage values.

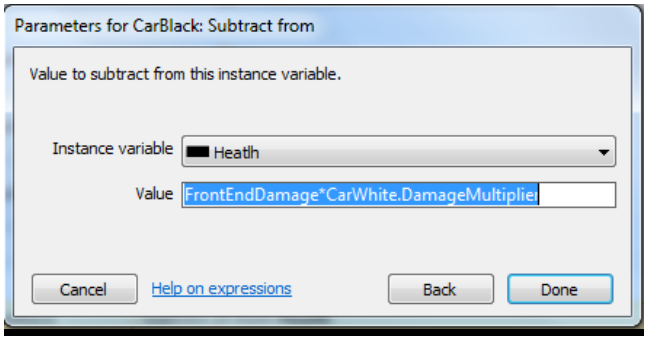
For now, you'll just add one variable for the front end damage.

In your event sheet, add a new global variable named **FrontEndDamage** and set its value to **25**.



Next go the the CarBlack on collision with CarWhite event. In its sub-event double click on the action that subtracts Health from CarBlack to modify the value. Change the value to **FrontEndDamage*CarWhite.DamageMultiplier**.

This will set damage CarBlack receives proportional to the speed of CarWhite.



Change the action in the other collision event that subtracts health from CarWhite to a value of **FrontEndDamage*CarBlack.DamageMultiplier**.

Run your layout to test your multipliers. You'll see the health removed will no longer be 25 every time, but rather a value based on the speed of the cars.

QUIZ TIME

Using the formula you set earlier (FrontEndDamage*CarWhite.DamageMultiplier). How much damage would be inflicted by a car traveling 100 pixels/second?

- 7.14
- 25
- 12.51
- 7

SUCCESS!

YOU'VE JUST FINISHED THIS SECTION.