Chapter No. 6
"Space Defender Part 1"
In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.6 "Space Defender Part 1"

A synopsis of the book’s content

Information on where to buy this book

About the Author

Miguel DeQuadros is a Game Developer and founder of the independent development studio Wurd Industries, based in Ontario, Canada. He has been developing iPhone games since the release of the App Store back at the exciting release of iOS 2.0. Since then, working under Wurd Industries, he has had eight games and one entertainment app published world-wide on the App Store, with more to come.

Originally interested in 3D animation and graphic design, Miguel focused mainly on this, but then got the game development "bug" and has been developing iPhone Apps since 2008. This allows him to use his creativity and knowledge of 3D animation for cut scenes and videos within his apps, and he loves every minute of it. Starting from his first project "Toy Tennis" back in 2008 (which, to this day, still remains a very popular app, thought it has been updated quite a bit with new graphics and social integration), down to his current project iMMUNE 3: The Final Stand, he continues to develop iPhone apps.

He began expressing interest in GameSalad when it was first launched in March 2009, but never started using it until January of 2010, when he started to develop his third iOS game, iMMUNE 2: Rise of the Salmonella, a 2D platformer sequel to the iMMUNE series. To this day, he and Wurd Industries continue to be Professional Members of the GameSalad community and still use GameSalad for prototyping and developing games for the iOS devices, and are now branching out to Mac development.

For More Information:

I would like to thank my dad John, my brother, and sister-in-law Johnny and Katie, my cousin Corey, my best friends Brandon and Kaleb, and a very special thanks to my wonderful fiance Joanne for encouraging me during the writing of this book and spending countless hours on the phone with me. There are many other of my friends who were very encouraging to me and I thank you for the constant boost of confidence. Without you guys I don’t think I could have done this book by myself. I would also like to thank Packt Publishing, and all the wonderful employees who helped me out, both in the initial stages and into the final chapters of the book. Your professionalism and ability to clearly explain things also helped me a lot, thank you Rashmi Phadnis, Srimoyee Ghosal, Kushal Bhardwaj, Priya Mukherji, Vishal D'souza, Hyacintha D'souza, and Newton Sequeira for everything. You made the production of this book a very smooth process.
GameSalad Beginner's Guide

In July 2008, Apple revolutionized mobile entertainment. They brought us the App Store. This was not only incredible for those who owned "iDevices" (iPod Touch, iPhone, and so on), who could now download apps and games to make their device even more entertaining, but it opened up a new scope for developers. The incredible new technology of the iPhone and iPod Touch made it possible for developers to create apps and games in which they could incorporate multi-touch controls, make use of the accelerometer and even allow the user's music to be played in-game. To date, the Apple App Store has over 425,000 apps available worldwide, and Apple has paid out billions of dollars to developers.

Apple also made it easy for individuals to become developers; anyone with programming knowledge and a Mac could become a developer, create apps and have their creation released around the world. The SDK (XCode) gave users the ability to create visually using the Interface Builder, and code everything within a uniform software development kit.

If you are like almost everyone else in the world, you don't know any programming language, nor do you have the time to sit down and learn one. That's where GameSalad comes in. What is GameSalad you ask? GameSalad is a fantastic, powerful creation tool that allows you to develop iPhone games! The best part about it is there is no programming involved whatsoever! With a simple drag-and-drop user interface, and behavior-based development, putting together an awesome game has never been easier! It is possible to create a fully playable game in less than half an hour! This book is going to cover the ins and outs of game development, sprite creation, planning, and all you need to know about GameSalad.

What This Book Covers

Chapter 1, You Need A Plan!, teaches you the basics of planning a game. You will learn how to think of an original idea, use storyboards, and create design documents, which will streamline your development phase by planning your game.

Chapter 2, Getting Started with GameSalad, guides you through GameSalad's user interface, creating a new project, working with actors and adding simple controls to a game.

Chapter 3, Add Zest to your Salad, teaches you how to create sprites (or images) for your game. Then we will import those images and some sound effects to add some more "zest" to our games.

Chapter 4, Starting Simple: Ball Drop Part 1, starts you off by creating a fully-fledged game, with physics, a menu system, and touch controls.

For More Information:  
Chapter 5, Starting Simple: Ball Drop Part 2, continues the creation of our ball drop game. By the end of this chapter, you will have created your first game in GameSalad!

Chapter 6, Space Defender Part 1, creates an Asteroids clone. We are going to create a good user interface, create smart enemies, and more!

Chapter 7, Space Defender Part 2, completes our Asteroids clone by setting up Game Center leaderboards. We will also test our game on our iOS device.

Chapter 8, Metal Mech Part 1, explains the creation of a large game that will have multiple large levels, make full use of the iOS (can also be used for Android publishing) accelerometer, and intelligent AI. In this chapter, we are going to create our user interface, smart enemies that will detect you as you approach, and butt on/accelerometer controls.

Chapter 9, Metal Mech Part 2, continues to work on our Mech game. In this chapter we are will add sound effects, particle explosions, jet trails and smoking, burning wreckage.

Chapter 10, Metal Mech Part 3, covers the creation of our level bounds (so the player can't walk through obstacles), turrets, weapon overheating, scoring, mission briefing (for a cool-looking User Interface!) and we are going to create the accelerometer controls!

Chapter 11, Metal Mech Part 4, teaches us to prepare our game for the iOS App Store. We will create iAds for extra revenue, Game Center Leaderboards, and we will deploy our app on to our device for testing. Then, we will upload our game to the App Store.

Appendix A, Getting Started in iDevelopment, covers the technical aspect of iOS development. We will discuss creating a developer account, provisioning profiles, installing Xcode and the iOS SDK, and more.

How was your break? I hope it was good! Anyways, remember back in 1979 when times were easier? When the best game on the market was a simple 2D single-colored set of pixels flying around a CRT screen? Yes in 1979, Atari released arguably the most influential game in history, Asteroids. I never saw the ’70s or most of the ’80s but I played Asteroids, as has the majority of the world and it is quite an addictive game. It’s one of those great games where you and a group of friends would play for hours to see who could get the high score. What does that have to do with this chapter? We are going to make an Asteroids clone, but we are going to have some better-looking effects. After all, our game is going to be on the iPhone!

This game is going to be a little more complex than our last one. We are going to incorporate some more touch controls via buttons to control the ship, include various enemies, and at the end of this chapter we are going to deploy our game to the iPhone for testing purposes, which will help immensely when getting your game ready to deploy to the App Store. You will get to see how it performs and controls on the device. Nothing would be worse than releasing a game that doesn’t work! Let’s see what we are going to cover in this chapter:

- Creating a good user interface and sprites
- Numerous AI (enemies)
- Sound effects
- Spaceship movement

Things are going to get juicy in this chapter, and by the end, you are going to have a great idea of how to effectively use GameSalad’s more complex procedures. As with the last game, this is going to be divided into two chapters because there is a lot to cover and you need to breathe in between. Let’s go!
Creating UI and sprites

I think, for this chapter, I am going to use Photoshop to design all the sprites. Let's start with the **Logo Screen**; this is simply going to be a screen that shows your company logo or a logo to represent yourself. I have a company, so following is the logo that I'm going to use for this game.

If the images imported into Game Salad are smaller than your actual image (for example, you import an image at 960x640 and it imports at 240x160) then, while saving your image, make sure your resolution is 72DPI that is, 72 dots per inch. Then, import it into GameSalad and you will have the correct resolution.

Ooh... pretty! This is exactly what the player wants to see! The previous screenshot is a flashy, cool-looking graphic, but don't forget that it has to match your gameplay. Ok, so let's import that into GameSalad, create a new project, and fill in all the details on the main screen.

1. Now, open up your **Initial Scene** and start editing it.
2. Import your image into the **Library** and drag it into your scene. All we are going to do with this actor is add a **Timer** behavior, so double-click it (the actor), drag in a **Timer** behavior and change its attribute to **Every 4 seconds**, and drag in a **Change Scene** behavior and set it to **Next Scene**.

![Timer and Change Scene Behaviors](image)

3. If you want to have more than one of these splash screens, then by all means do it; I did it too.

![MIGapps splash screen](image)
4. Just copy the first level, change the image, and keep the timer the same. Now, let’s move on to our Main Menu; create a super flashy looking background for this one. As for me, I am going to imitate the original Asteroids box art.

It's pretty easy to create splash screens, isn't it?

Creating the menu and more sprites

Now that we are done with our company logos, we are going to create the menu.

1. Add a new actor, and name it Text Button or something similar. Drag it into the middle of the level and double-click to edit it.

2. Add a new rule to the actor, change it to Actor receives event | touch | is | pressed; then add a Change Scene behavior to the rule and change its setting to Next Scene.

3. Then add a Display Text behavior; and change the text to Tap to Play. For now, this is going to be the only button we add; but in the next chapter, we will add some more.

4. Create a new level and name it Level, this is going to be our playground for our spaceship to save the world. Next, we have to design the sprites:
Our background: Following is a cool background screenshot for our level, lots of stars, and some space dust floating around.

The asteroid: Following is a screenshot for the asteroid, nothing too fancy but it still looks like an asteroid.

Player spaceship: Following is a screenshot for your spaceship, it looks pretty cool, doesn’t it? If you want to make it look better, you can!

Enemy spaceships: Finally, the enemy spaceship, just a classy looking flying saucer.

For More Information:  
Have a go hero

Why not try designing your own sprites? Make them look as cool as you like, cartoony or realistic, it’s all up to you!

Making Level 1

This first level is going to be an easy level for the player. We are going to arrange our sprites into the scene. This is only our first level, so we will only include two asteroids in this level, adding more in later levels as the game goes on.

In the previous screenshot, you can see the level put together with two asteroids. After I did this, I duplicated the Big Asteroid actor in the Inspector, renamed it Small Asteroid and sized it down by half. So instead of 50 x 50 px, the Small Asteroid is 25 x 25 px. I created these Small Asteroid actors because when you shoot the Big Asteroids they will spawn three smaller asteroids like the original game. Cool huh?

1. Now, we are going to create a score actor. So, create a new actor and name it Score, open it up and change the Width to 100 and Height to 20.

2. Go back to the level editor, click the Attributes tab and click the + button to create a new attribute. When the selector comes up, choose Integer, and rename it to Score.

3. Now, go back to the Actors tab and double-click the Score actor, so we can add in some behaviors.

For More Information:
4. Add a **Display Text** behavior, and click the e beside the textbox, then click the down arrow to expand the **Expression Editor** and click **Game** and then **Score**, or you can even type in `game.Score`, and then click the green arrow to accept the changes. Change the size to 15, and keep the color white.

5. Now, we have to change the alpha of the actor, but we are going to hide it when the game starts to make it easier to arrange in the level editor.

6. All we have to do is drag in a **Change Attribute** behavior, change the `self.Color.Alpha` to 0.

![Change Attribute](image)

7. Test the level to make sure it works and disappears when you play. If not, go back to double-check the behaviors and see if you typed everything in properly.

8. Now, we have to edit the asteroids. Double-click the **Big Asteroid**, so we can edit it. Add in a **Rotate** behavior and set the **Speed** to `random(-50,50)`, so this will make the asteroids rotate at different speeds. The negative number will make it rotate counter-clockwise and the positive number will make it rotate clockwise.

9. Now, add in a **Move** behavior, and change the **Direction** to `random(0,360)`, then change the **Speed** to `random(5,50)`. This will make the asteroid move in a random direction from 0 to 360 degrees, and then the speed will make it move slow or fast.

![Asteroids](image)
What did we do? Just to recap, we added custom behaviors to randomize the movement of these asteroids; we varied the rotation, the speed, and the direction they move in with simple behaviors.

**Time for action – player collisions**

Now, we have to add in the collisions and explosions.

1. Let's edit our player. Double-click the **Player** actor in the **Inspector**. Add a rule, change it to **Actor receives event | overlaps or collides | with actor of type | Big Asteroid**. Now, add two **Particles** behaviors, on the particle event (this is going to be the shrapnel) and change the following settings:
   - **Spawn Rate**:
     - **Number of Particles**: 20
     - **Particles Startup Time**: 1
     - **Particle Lifetime**: 2
   - **Velocity/Position**:
     - **Direction**: random(0,360)
     - **Speed**: random(20,100)
   - **Size**:
     - **Size**: random(2,15)
   - **Color**:
     - **White Color Does not Change**
   - **Rotation**:
     - **Angular Velocity**: 50

For More Information:
Chapter 6

- Image:
  - Draw an image for the flying shrapnel.

The previous image is the one I drew. It has really sharp-jagged edges that make it look like it blew apart!

2. Now, for the second Particle Event, change the following settings:

- Spawn Rate:
  - Number of Particles: 4
  - Particles Startup Time: 0
  - Particle Lifetime: 2

- Velocity/Position:
  - Direction: 0
  - Speed: 0

- Size:
  - Size: 15 | Size changes to | Size Transiton
    - Target Size: 100
    - Duration: 0.3 seconds

- Color:
  - Color: Banana
  - Blending: Additive
  - Color Transiton | Target Color: Black
  - Duration: 1 second

For More Information:
Space Defender Part 1

- **Rotation:**
  - Angular Velocity: 50

- **Image:**
  Draw or download an explosion image, there are so many explosion sprites on the Web, just make sure it has a black background.

3. That looks pretty cool! Now, we have to destroy the ship and reset the level. Click the Attributes tab in the Inspector. Now, click the + button, choose a Boolean attribute and name it Dead?, or something similar.

4. Go back to editing the Player actor. Now, add a Change Attribute behavior and change it to game.Dead? | To: | true.

5. Now, add another rule and change it to Attribute | game.Dead? | is | true. Add a timer and change it to Every 1 second, then add a Destroy behavior.

For More Information:  
6. Next go back to the level editor and double-click the background image to edit it.

7. Create a rule and change it to **Attribute | game.Dead? | is | true**; add a **Timer** and change it to **Every 3 seconds**, then add a **Reset Scene** behavior to the Timer, and change its attribute to **Change Attribute: | game.Dead? | To: | false**, as shown in the previous screenshot.

You have to be very careful about the previous one. The **Change Attribute** behavior **MUST BE** below the **Reset Scene** behavior. This way the level will reset before the attribute is changed, if it was the other way around the attribute would be changed, and the level wouldn’t reset because the Rule is no longer true.

Test it to make sure it works, if not, go back and check all your behaviors to make sure everything was typed in right.
**Space Defender Part 1**

**What just happened?**

With some more complex behaviors, we were able to call some behaviors after a collision; we changed an attribute on a collision, created some awesome-looking particle effects, and reset the level all due to that one collision.

**Time for action – making the asteroids go kaboom!**

Now, we are going to do the behaviors for the asteroids, such that, when you shoot at them, they break into three smaller asteroids. First off, you have to create the laser image.

1. Double-click our Level 1, so we can edit it. Now, import that image into our game and create a new actor from it and name it Laser.

2. Double-click the Big Asteroid actor, so we can edit it. Create a new rule and name it Destroy or something easy to remember.

3. Change the rule settings to Actor Receives Event | overlaps or collides | with Actor of type | Laser. Now, add in three Spawn Actor behaviors, changing the Actor setting for each one to Small Asteroid, then drag in a Destroy behavior, as shown in the following screenshot:

4. Now, we have to edit the Small Asteroid actor, but before we go to edit it, copy the two Rotate and Move behaviors from the Big Asteroid, so we can paste them into the Small Asteroid just to save some time.

---

For More Information:

You can save even more time by saving that behavior as a custom behavior. Don’t know how? In the Library, click the Custom tab, and drag your behavior into the custom box.

5. Double-click the Small Asteroid and paste these behaviors in. Now, we have to add in a new rule, change it to Actor Receives Event | overlaps or collides | with Actor of type | Laser, then add in a Destroy behavior.

6. Now, we have to work on the laser. Double-click to edit it; change the Blending mode to Additive, to give it a glowy look.

7. We now have to add in a few rules, one for the Big Asteroid, and another for the Small Asteroid. Create a rule and let’s start with the Big Asteroid, change it to Actor receives event | overlaps or collides | with actor of type | Big Asteroid.

8. Add in a Destroy behavior, then drag in a Change Attribute behavior. Change the attribute from game.Score to game.Score + 10. This will add 10 points to the score each time you hit the asteroid. Do the same for the Small Asteroid.

What just happened?

We were able to set up a simple scoring system that will be used throughout each level, no need to set up separate Score attributes per level. This is a global attribute that can be accessed throughout the game. Later, we are going to look at uploading our score to Game Center.

For More Information:


**Time for action – creating the sound effects! (pew pew kaboom)**

Ok, we need to add a little more juice to this game, sound effects! No game is complete without them! Let’s start off with playing a song at the beginning of our game, right at the splash screens. Create a song in GarageBand, if you have it, because then you can use it royalty-free, but if you can’t, download some indie music, or royalty-free sound effects/songs (keep in mind you only have to do this, if you are planning on making money; if you aren’t going to make money, you can use any song you want). I’m going to use something retro, "The Final Countdown".

1. When you are importing the song, GameSalad will ask you whether you want to import the song as a **music file** or a **sound file**. Of course, for this, we will select **music file**. Let's go to our first level, which is our splash screen, and double-click the actor that is displaying your logo. Then simply drag the song into the behaviors area. When you test your game, the music will start playing.

2. Now, let’s add in the sounds for the asteroids being destroyed. Let’s go to our **Inspector** and double-click the **Big Asteroid** actor. Download an explosion sound and import it into GameSalad. Once you have done that, drag the sound effect into the **Destroy** rule. Alternatively, you can drag in a **Play Sound** behavior then select the sound effect you want, but I find it quicker just to drag the sound effect into the rule you want.

For More Information:

3. Do the same for the **Small Asteroid**.

4. Now, for the **Player**, double-click it. Download a different explosion sound effect, something destructive and sounds like there is shrapnel flying around. We want this to be somewhat realistic right? If you don't want to, you can always set a car horn sound effect for the player exploding, that will make the player laugh. Do whatever you want, make it your awesome masterpiece.

5. Where was I? Oh yes... the player dying. Ok, so all we have to do is drag our sound effect into the **Die** rule and watch, or should I say listen to the magic! Also, if you want, you can play a failed sound effect to find out where you put the rule that will reset the level after the player dies; in our case, we put it in the **Background** actor. Drag in the sound, or music file that you want in the **Timer** within the **Rule**, this will play the sound just before the level resets.

---

For More Information:
Have a go hero

Have a try! Find your own sound effects and music files for these rules, make them as realistic or hilarious as you like! Whatever you feel would be most appealing to the player.

Creating spaceship movement

Now, we are going to focus on the movement of our player. There are a few ways that we could use to set up the controls, we could do touch or tilt controls, but we are going to use touch controls for rotating, and tilt controls for accelerating. Let's start with the acceleration.

Time for action – player acceleration

Now for the fun bit, and the most critical... the player movement!

1. Double-click the Player actor to edit it. Click the Create Group button, and name it Movement. Make sure Group is selected; when it is, you can see a blue outline around it. If it isn't, select it. Now, click the Create Rule button and you should see a new rule within your group. If not, simply drag the new rule into it.

2. Rename the rule to Acceleration and change the settings to: Attribute | game. Accelerometer.X | > | 0.5. This says that when the accelerometer on the X axis is greater than 0.5g, which means the device is in landscape and tilted forward.

3. Now, add an Acceleration behavior, change the Direction to 90, and leave Acceleration at 100. Now would be a good time to limit the speed the player can get to, in the actors attributes box; expand the Motion roll out, change the Max Speed to 200, and check the Apply Max Speed box.
What just happened?

We were able to utilize the power of GameSalad and the iPhone to use the device's accelerometer to accelerate our player in one simple rule! GameSalad just keeps getting easier!

Time for action – player rotation buttons

Now, we need to go back to the Scene Editor. In the Inspector, we need to create two new actors; name one Left and the other Right. These will be the buttons that will rotate our player.

I used an image of a joystick thumb pad for my buttons in the game, but you can use any image you like.

1. In the Attributes tab, create a new attribute, select Integer, and then name it Rotation. Now, go back to the Actors tab and double-click our Left actor.
2. Change the Width and Height to 50, and then in the Color roll-out change the Alpha to something barely visible, so .5 or less would be perfect. Just make sure the player can see it, as shown in the following screenshot:

![Screenshot showing the Attributes panel with Width and Height set to 50, Alpha set to 0.3, and other settings]

3. Now, let's create a new rule, and name it Pressed, then change the rule settings to Actor Receives Event | touch | is | pressed.

4. Then add a Change Attribute behavior into the rule, and change its settings to: Change Attribute: | game.Rotation | To: | 1.

5. Expand the Otherwise roll-out and copy the Change Attribute behavior in there and change it to 0 instead of 1.

6. Now, go to the Right actor and do the same. Change its Width and Height to 50, the Alpha to something less than .5 and repeat the rule, name it Pressed, then change the rule settings to Actor Receives Event | touch | is | pressed.

7. Then add a Change Attribute behavior into the rule, and change its settings to: Change Attribute: | game.Rotation | To: | 2.

8. Expand the Otherwise roll-out and copy the Change Attribute behavior in it, and change it to 0 instead of 2.

For More Information:
What just happened?

In this rule, when the player touches the button it will change the Rotation attribute to 1, then the Otherwise roll-out will change it to 0 when the player isn't touching it. Thus, it stops the actor from rotating once the player isn't touching the button.

Time for action – rotating the player

Now, we have to go back and edit the Player actor. Double-click it in the Inspector, so we can make some changes.

1. Create two rules in our Player actor, name them Left and Right and drag them into our Movement group. We are putting all our movement behaviors into one group to make things a lot less cluttered, because, trust me, when you start making a game with numerous complex behaviors, things can get a little messy. So putting all relevant behaviors into one group is just a smart thing to do.

2. Change the Left rule to Attribute | game.Rotation | = | 1

For More Information:
3. Then drag in a Rotate behavior, change it to Counter-clockwise and leave the Speed at 90.

4. Do the same for the Right rule:
   - Attribute | game.Rotation | = | 2
   - Rotate behavior Clockwise Speed 90

Run your level to see if it works.

Sure enough it does!

Now, I want to make the UI look a little better, I want a nicer-looking HUD so I’m going to design one.
Now, let's import that into GameSalad to see how it looks...
It looks pretty good! I added an **Additive** blending option to it to make the green outlines glow like a computer screen. Awesome!

By now, you must love GameSalad because it is easy to use. Plus, it's so powerful! We haven't even gotten into some of the really complex things, but GameSalad makes all that so simple!

In the previous screenshot, as you can see, I am actually testing the game on my device.

In the next chapter, we are going to look at testing for a particular using the GameSalad Viewer, as well as actually deploying the game onto your device. We are also going to look at wrapping the actors around the screen (when the actor reaches the edge of the screen it will wrap around to the opposite end of the screen, so it doesn't disappear), shooting your lasers, particle boosters to make the game look better, Game Center leaderboards, and more levels including ones with alien ships! Are you excited to get into it? Me too! Let’s take a look at what it's going to look like first, just to whet your appetite...

---

**For More Information:**
Ooh! It's giving me the chills, it looks so much fun! Are you eager to get to it now? Well, go ahead! Turn to the next chapter and I'll see you there!

**Pop quiz**

1. How do you detect forward tilt acceleration?
   a. Add a **Move** behavior with a direction of `game.Accelerometer.X`
   b. Create a rule that has an attribute of `game.Accelerometer.X`
   c. All of the above
   d. None of the above

2. What does the change in the **Move** behavior's **Direction** to `Random(0, 50)` and **Speed** to `Random(5, 50)` do?
   a. Moves the actor at a random direction and speed
   b. Wraps the actor around the screen
   c. Shoots particles at a random speed
   d. None of the above

For More Information:
3. How do you spawn more than one actor?
   a. Drag a Spawn Actor behaviors and select Number then specify the number
   b. Drag a Spawn Actor behavior per number of actors you want to spawn
   c. You can’t, only one actor can be spawned at one time
   d. None of the above

Summary

In this chapter, we started to create a good-looking game, we designed all the sprites to look really cool, then we started making some enemies (the asteroids). We added some zest into our "salad" by adding some sound effects, music, splash screens, and then to top it off, we looked at how to create some realistic spaceship movements.

For More Information:
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.