W. P. CAREY
SCHOOL *of* BUSINESS
ARIZONA STATE UNIVERSITY

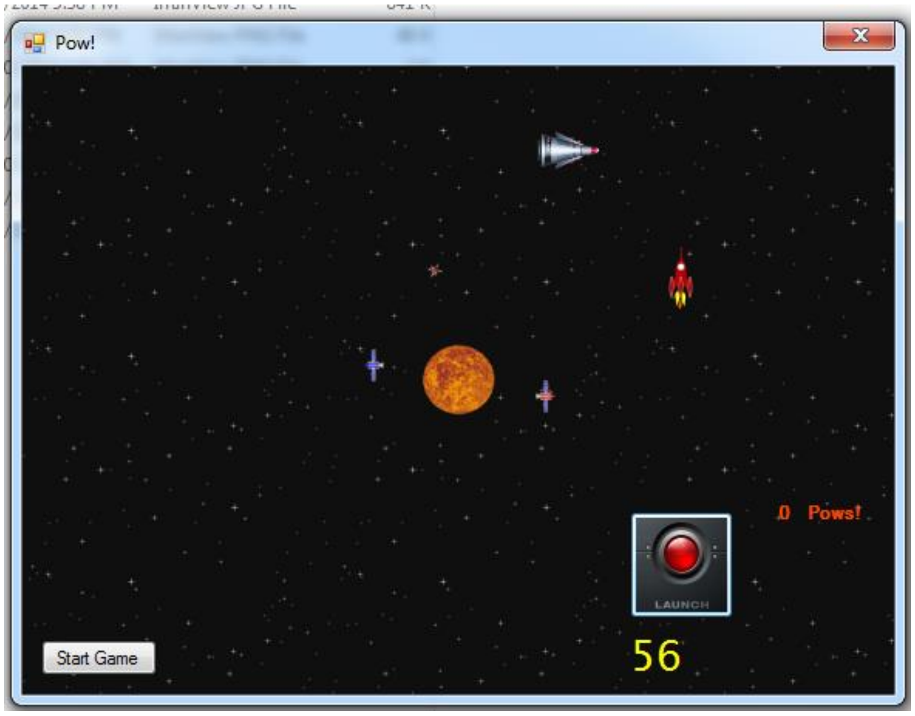
Assignment 9: Make-up Assignment

*This assignment is **optional**. If you complete it, your lowest assignment score will be dropped. If your makeup assignment score is lowest, it will be dropped. If you don't do this makeup assignment, Assignment 9 will be "dropped" while your first eight assignment scores will be taken into account.*

Due on: Friday, December 5, at 10 PM to Blackboard

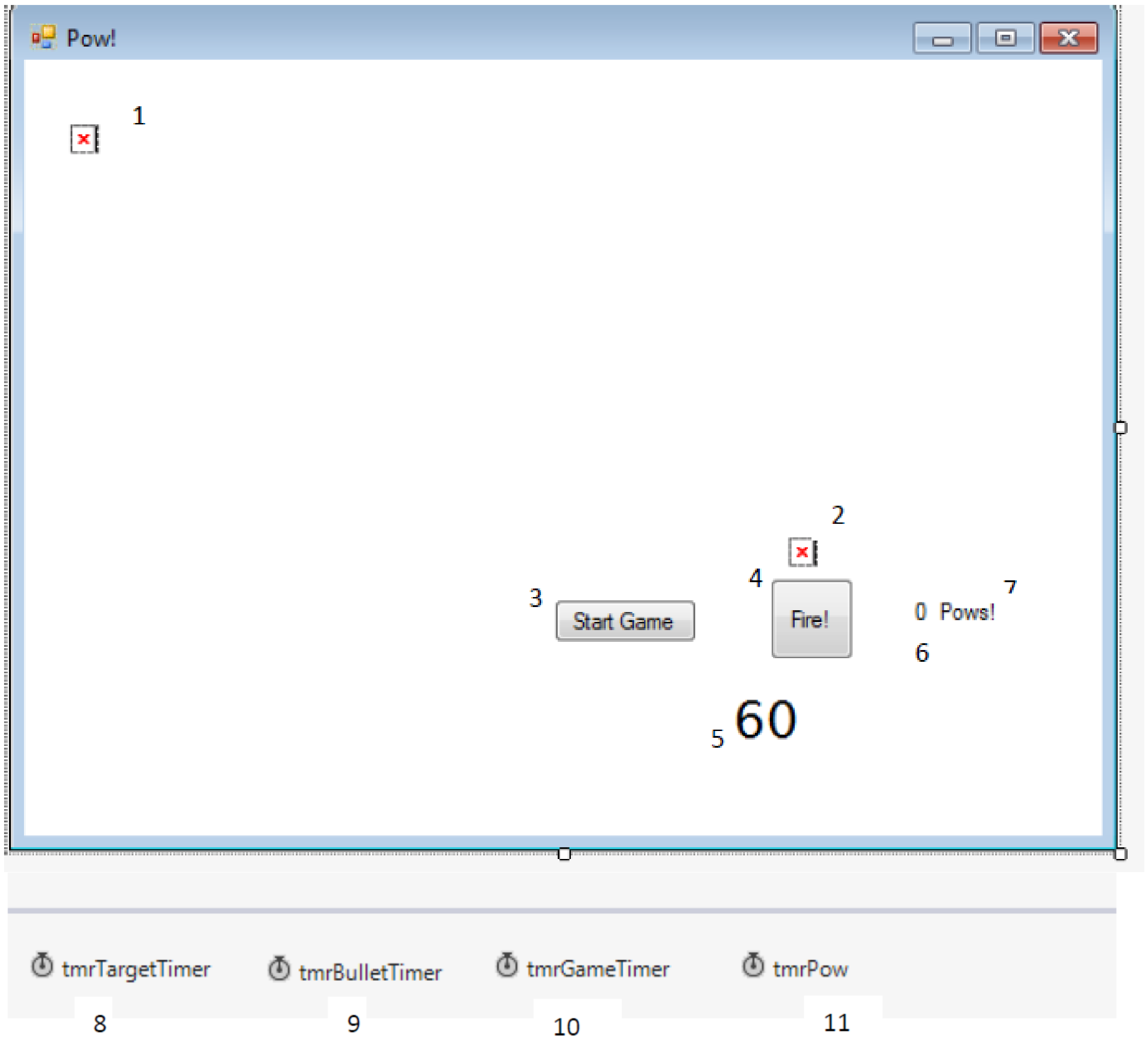
Create a game based on the in-class work we did to move labels dynamically at runtime. Implement the *Pow!* game, which has target bullet moving at a randomized speed. The objective is to fire and manipulate another moving bullet so that the two bullets collide – *Pow!* The game is timed for 60 seconds.

Skills developed: Utilizing Visual Studio Form applications, declaring and instantiating controls, manipulating controls dynamically, working with event handlers, picture boxes, building moderately sophisticated logic.



Project Details and Requirements
A sample executable file is uploaded and available on Blackboard for you to test program behavior.

Project Name: Pow
Form Name: Pow



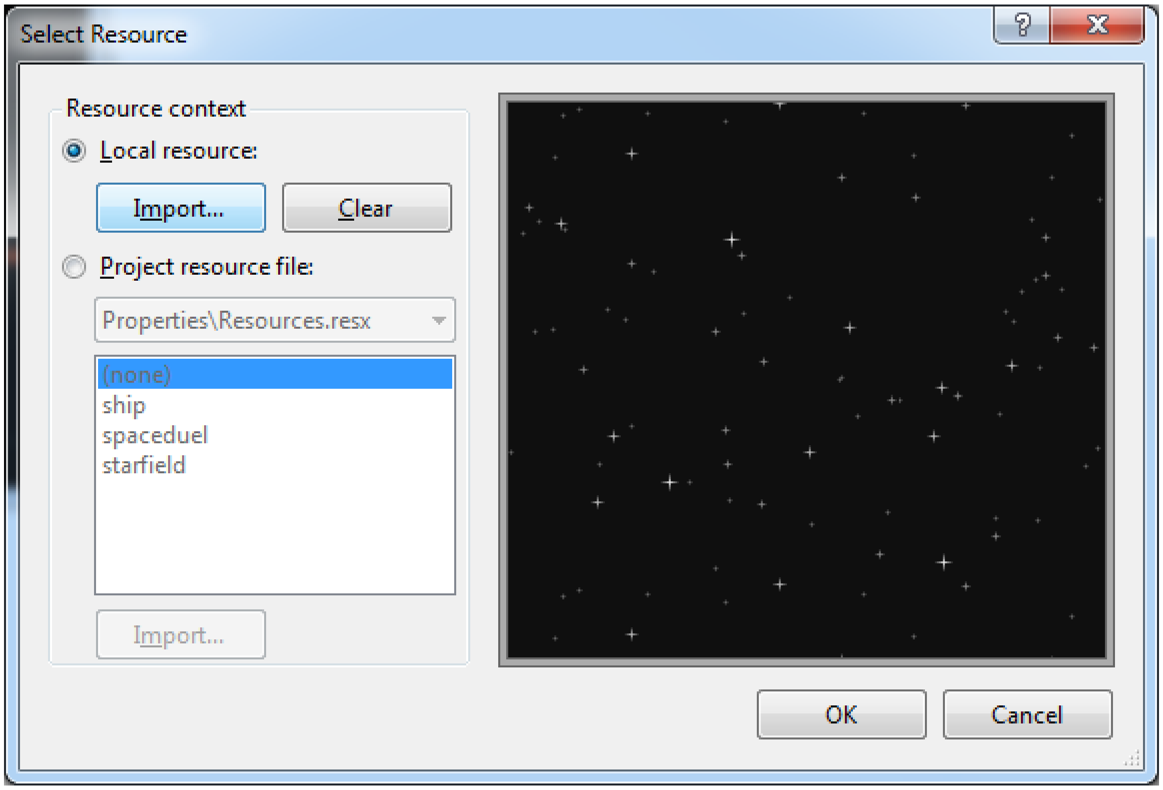
***Controls**: Note the basic design of the form in the picture above. There are 11 controls marked with numbers. Recreate the form with all of the controls. Adjust location of controls as needed.*

Use suffixes within the control names so that when you work with in code, you can easily distinguish a TextBox from a Label. e.g. “scoreTextBox” should easily be interpreted as the TextBox which contains a score, while “scoreLabel” should easily suggest that is the label accompanying a score.

Important - Copy all provided media files (.png, .jpg and .wav) to the bin\debug folder within your project <u>before</u> you start configuring control properties.

Form Properties:

- Name: Pow
- Text: "Pow!"
- FormBorderStyle: FixedSingle
- MaximizeBox: False
- MinimizeBox: False
- Size: 580, 444 *(if your screen resolution needs a different size, adjust the size)*
- StartPosition: CenterScreen
- BackColor: White
- Background Image: Import space.png from "Local resource." (see picture below)
- BackgroundImageLayout: Strech



Using the IDE, create the following controls.

1) PictureBox, targetPictureBox, to keep show the target (the ship that runs horizontally).
   Location: 24, 34. Size: 40, 40.
   Image: ship.png (import local resource)
   SizeMode: StretchImage
   BackColor: Transparent

2) PictureBox, bulletPictureBox, to show the fired bullet (the rocket that runs vertically).
   Location: 416, 236. Size: 40, 40.
   Image: rocket.png (import local resource)
   SizeMode: StretchImage
   BackColor: Transparent

3) Button, startButton, to start the game.
4) Button, fireButton, to fire the bullet.
   BackgroundImage: Launch.jpg
   BackgroundImageLayout: Stretch
5) Label, statusLabel, to show the countdown timer. (Set Font/ForeColor as desired).
6) Label, powPrefixLabel, to show how many collisions/pows.
7) Label, powSuffixLabel to list the text "Pows!"

8-11) Timer controls. There are 4 timers. Name them: targetTimer, bulletTimer, gameTimer, and powTimer. Each timer controls a different timed event. The target timer controls the movement of the target bullet (the ship at the top left). The bullet timer controls the movement of the fired bullet (the rocket at the bottom right). The game timer keeps a track of the time. The Pow timer controls the animation of the collision between the two space ships/rockets.

Change the Text and Location properties using the IDE to reflect the form controls as shown. ***All the important properties are listed here. Fill in any other needed ones on your own! Most of the controls ought to be invisible (set the Visible property) at loading time. The images ought to be stretched, the background ought to be transparent. Feel free to manipulate fonts, colors, size, location, etc. for all controls to suit your screen and liking.***

Your form can look somewhat different than the template shown here. Be sure to make any adjustments to your program to make it run properly if you choose to modify anything from what is listed here.

Pow.cs Partial Class

Do NOT add any code in the Designer.cs or Program.cs classes.

1) Within the Form Pow.cs partial class, add the following ***instance*** variables.

1 Random variable, r, to randomize speed

integer, targetSpeed, to store speed of moving target.

integer, totalScore, to store user score.

integer, secondsLeft, to store game time left.

SoundPlayer, soundPlayer, to play music. (To use The SoundPlayer class, add a "using System.Media" statement at the top of the .cs file)

METHODS:

Methods should go inside the class as with all objects. The precise location (top, bottom, etc.) does not matter so far as methods are located within the class block.

2) StartButton Click event hander.

Purpose: The purpose of this event handler is to start the game and reset all the properties to an initial state. It then starts the game by starting the timers.

General Method Logic:

- Instantiate your random object, r.
- Generate a new random number from 4 to 15 to set the speed of the target bullet. Store this newly generated number in targetSpeed.
- Set the Interval property of the Target Timer to 20 milliseconds.
- Set the Interval property of the Game Timer to 1000 milliseconds.
- Set the Interval property of the Bullet Timer to 10 milliseconds.
- Set targetPictureBox's Left to 0.
- Set the bulletPictureBox's Left to 236.
- Set totalScore to 0.
    - Set the PowPrefix label to show totalScore in it. You will need to Convert totalScore to a string and then assign it to the Text property.
- Set the following controls' Visible property to true: statusLabel, targetPictureBox, bulletPictureBox, powSuffixLabel, powPrefixLabel, fireButton.
- Set fireButton's Enabled property to true.
- Set secondsLeft to 60.
    - Set the statusLabel label to show secondsLeft in it. You will need to Convert secondsLeft to a string and then assign it to the Text property.
- Call the Start method of the Target Timer.
- Call the Start method of the Game Timer.

3) targetTimer_Tick event handler.

Create the Tick event handler for the targetTimer timer.

Purpose: This method moves the target bullet to the right. When the bullet comes on the screen from the left, it will select a new random speed.

General Method Logic:

- Increment the Left of targetPictureBox by targetSpeed i.e. not by 1 but rather use the number stored in targetSpeed.
- If the Left of targetPictureBox is greater than or equal to the Width of the Form (use the "this" keyword to get the Width property of the form)
    - Rest the Location of the targetPicture box. For this, you will have to create a new instance of Point and supply the constructor with the x and y coordinates (0 and 34). Store the reference to the newly created Point object in Location (which accepts a Point).
    - Generate a new speed by using the Random object. Use 4 and 15 as the min and max.

4) gameTimer_Tick event handler.

Create the Tick event handler for the gameTimer.

Purpose: This method updates the game timer by changing the label text and updating the secondsLeft variable.

General Method Logic:

- Decrement secondsLeft by 1.
- If the secondsLeft are greater than 0
    - Show the seconds left in the statusLabel Text property. You will need to convert the integer to a string.
- Else
    - Stop the Game Timer, since there is no more time left in the game. Call the Stop method of the Game Timer object.
    - Stop the Bullet Timer. Again, call the Stop method.
    - Stop the Target Timer, again, by calling the Stop method.
    - Disable the Fire button by setting its Enabled property to false.
    - Set the Status label's Text property to "Game over"

5) fireButton_Click event handler

Create the Click event handler for the fireButton button.
Purpose: This method fires the bullet by setting starting up the appropriate Bullet timer.

General Method Logic:

- Set the Visible property of bulletPictureBox to true
- Start the Bullet Timer by calling its Start method.

6) bulletTimer_Tick event handler.

Create the Tick event handler for the bulletTimer timer.

Purpose: This method sets what happens at every tick event for the bullet that was fired. Accordingly, it will move the bullet as shown by the bulletPictureBox image. Then it will call the Collided method to test if there has been a collision. If yes, it start the mechanism to show a collision. Finally, it will ensure that once the bullet goes above the visible portion of the form, it comes back to its original location.

General Method Logic:

- Decrement bulletPictureBox's Top property by 7
- Within an if statement condition, call the Collided method (write the method in step 7) If Collided() returned true,
    - Increment totalScore by 1. *It was a hit!*
    - Update the powPrefixLabel label with the new totalScore. Remember to convert!
    - Stop the Bullet Timer
    - Stop the Target Timer
    - Change the ImageLocation property of targetPictureBox to "pow.png". This will change the picture to the pow picture.
    - Set the Height and Width of targetPictureBox to zero.
    - Set the Visible property of bulletPictureBox to false. This will hide the bullet, which has already collided with the target.
    - Call the Start method of the powTimer object. That will animate the pow picture.
- If the Top of bulletPictureBox is less than or equal to zero
    - Stop the Bullet Timer
    - Set the Visible property of bulletPictureBox to true
    - Set the Top property of bulletPictureBox to 250. This restores it to its original location.

7) Collided method.

Return parameter: bool

Purpose: This method tests if the target bullet and the fired bullet have collided. It will do that by testing the top and left properties of both the controls. If it detects an overlap, it returns true.
General Method Logic:

```
// check for overlapping controls on the left side of bulletPictureBox.

IF ((bulletPictureBox's Left is greater than or equal to targetPictureBox's Left AND the
bulletPictureBox's Left is less than or equal than targetPictureBox's Right)

AND

(bulletPictureBox's Top is less than or equal to the Bottom of targetPictureBox AND
bulletPictureBox's Top is greater than or equal to targetPictureBox's Top)),

THEN
    return true.

// the above covers only one side of the bullet. The same needs to be done for the
other side below.


IF ((bulletPictureBox's Right is greater than or equal to targetPictureBox's Left AND
bulletPictureBox's Right is less than or equal to targetPictureBox's Right)

AND

(bulletPictureBox's Top is less than targetPictureBox's Bottom AND
bulletPictureBox's Top is greater than or equal to targetPictureBox's Top)),

THEN

return true

ELSE

return false.
```

8) powTimer_Tick event handler.

Create the Tick event handler for the powTimer timer.

Purpose: This method adds a rudimentary animation to show that the two bullets collided by displaying the Pow image.

General Method Logic:

- Incremenent targetPictureBox's Height and Width by 1 each.
- If targetPictureBox's Width is equal to 25
    - Stop powTimer by calling its Stop method.
    - Set the Top of bulletPictureBox to 250.
    - Set bulletPictureBox to visible by using its Visible property.
    - Set targetPictureBox's Height and Width to 40 each.
    - Rest the Location of the targetPicture box. For this, you will have to create a new instance of Point and supply the constructor with the x and y coordinates (0 and 34). Store the reference to the newly created Point object in Location (which accepts a Point).
    - Set targetPictureBox's ImageLocation to "ship.png"
    - Generate a new random number from 4 to 15 to set the speed of the target bullet. Store this newly generated number in targetSpeed.
    - Start the TargetTimer by calling its Start method.

9) Form Load event handler

Create the Load event handler for the Form

Purpose: This method is executed before the Window is displayed to the user. It will create the environment for the looping sound file.

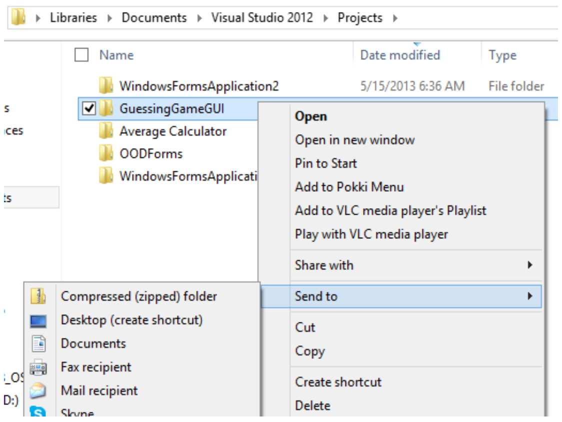
General Method Logic:

- Create a new instance of SoundPlayer and supply the constructor with the name of the wave file (s2.wav) as a string. Remember to assign it to your instance variable.
- Call the Load method of your soundPlayer instance.
- Call the PlayLooping method of your soundPlayer instance.

Submission Instructions

Submission should be made using a zip file that contains all of the Visual Studio C# project files. You will need to ***zip the entire project folder*** along with the .sln and .suo files. The folder will automatically contain the class source files as well as the executable file that is generated in \ProjectName\bin\Debug folder. Upload file to the Blackboard assignment drop box.

Zip the entire top-level folder by right-clicking the folder and selecting Send to | Compressed (zipped) folder.



Using built-in windows zip tools: http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files

Make sure you check the following. Your grade is dependent on all these criteria being met.

- You have included your name as a comment within your class.
    - e.g. " // Assignment 9, Jane C. Smith, CIS 345, Tuesday 9:00 AM"
- Zip filename is: FirstNameLastName_Assignment9.zip
- Your code is commented and you are using all prescribed programming conventions.
- Your code utilizes PascalCase and camelCase as appropriate.

Verify your zip file before you submit

- Check for actual class files being present in the folder before you zip it.
- Check your zip file size after zipping – if it under 4K, it likely contains only a shortcut.
- Uncompress your zip file before submitting and verify that files are present.
- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.
- Make sure you have submitted your file and not just saved a draft on Blackboard. A blue clock indicates a submission in progress, i.e. a draft, not a submission.

This takes an extra couple of minutes. Please do it if your grade is important to you. If you do this, you will not end up submitting a bad file. If you submit an empty file, or one containing only a shortcut, or a bad zip file, or a bad project file, you will receive a score of zero. ***This is the Makeup assignment, so there will be no making up the makeup assignment! If you submit a file that cannot be properly extracted and run, you will have NOT have another chance at doing the makeup..***