Overview

During this assignment, you will bring together your knowledge from the previous weeks and build on it to create an engaging, collision-based 2D animation. There are many different kinds of choices you may make with this work, so take time to explore your options before getting started. You should not feel limited by the possibilities provided here. Feel free to try out new and different ideas as you create!

Prompt

You will be completing your coding work in Visual Studio. Be sure to begin with a project file that has the libraries set up correctly from your earlier efforts in Module One. Then, copy the starter code from the Module Eight Assignment Code TXT file into your Visual Studio project. Remember, this is the same code you reviewed in the video for this week. From here you will be able to manipulate the provided code in a number of different ways. Note that Linmath.h is the only file from the original ZIP folders you used to complete your setup in Module One that is used in this week's provided code.

Specifically, you must address the following rubric criteria:

- **Arrange the bricks into an organizational structure that promotes engagement with the animation**. The goal is to create a layout that is visually unique or compelling and includes bricks of a style that makes the animation more interesting to watch. Some options for details you may wish to include are as follows, but you do not need to complete all of these. You can also try an idea of your own instead.
    - Add texture or color to the different kinds of bricks.
    - Change the sizes of the bricks.
    - Add a manually controlled paddle to the bottom of the screen, using the brick item as a base.
- **Apply physics laws to the circles**. When a circle hits one of the sides of the screen, its progress should be altered in some way. While the circles currently move at a constant speed and have randomized movement once they bounce off one edge of the screen, there are ways you can alter this to make the animation more engaging. Some options you may wish to use for your work are as follows, but you do not need to complete all of these. You can also try an idea of your own instead.
    - Alter the speed of the circle.
    - Change the angle of trajectory so it follows physics laws instead of taking a randomized pattern. (This means it would continue in the direction it was heading rather than moving backward.)
    - Add friction to specific surfaces, which would affect the circle and slow its progress once it collided with the surface.
- **Alter the state of the bricks upon collision**. When a circle collides with a brick, you will need to code for an event to occur. This means updating the code to alter the state of the bricks upon collision. Some options you may wish to use for your work are as follows, but you do not need to complete all of these. You can also try an idea of your own instead.
    - Require the brick to take a certain number of hits before it disappears.
    - Change the color or texture of the brick when it is hit.
    - Combine both of the previous two options, meaning the brick changes its texture each time it is hit until it disappears. For example, you may choose to add cracks to the texture of the brick until it is "destroyed."
- **Alter the state of the circles upon collision**. When a circle collides with another circle, you will need to code for an event to occur. This means updating the code to alter the state of the circles upon collision. Some options you may wish to use for your work are as follows, but you do not need to complete all of these. You can also try an idea of your own instead.
    - The two circles combine to become one larger circle.
    - The circles change their color or texture.
    - Both circles disappear once hit.
    - The circles spawn multiple smaller circles.
- **Explain the changes you made to the code**. Discuss the work you completed by focusing on the different tactics you used to create a fully realized 2D animation. What were the changes you chose to make? What was your intent behind them? How did you approach coding to successfully create this outcome?

- **Create code that follows a logical flow without syntax errors**. The code you create has to be executable and all the code that is included needs to be reached by the execution. Note that not everything should be written in a single function and your work should be well-modularized.
- **Apply coding best practices in your creations**. Pay particular attention to the way you format and comment your code. Program code should be easy to read and follow industry standard code formatting practices, such as indentation and spacing. Commenting best practices should be in place to ensure the code is briefly and clearly explained using descriptive comments.

Guidelines for Submission

Submit a 100- to 200-word Microsoft Word document along with a completed ZIP folder containing all of your code. Your ZIP folder may include one or multiple CPP files along with Visual Studio project files. It should also include an EXE file, because without this your code will not be able to run. Checking for the EXE can be used as a quick reference on the functionality of your code before you submit.

Module Eight Assignment Rubric

Criteria	Exemplary (100%)	Proficient (85%)	Needs Improvement (55%)	Not Evident (0%)	Value
Brick Arrangement	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner; areas demonstrating exemplary work may include completing a particularly complex arrangement of the bricks, adding texture, or creating a paddle	Arranges bricks into an organizational structure that promotes engagement with the animation	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include moving, adding, and otherwise altering the existing bricks at an appropriate level of complexity	Does not attempt criterion	15
Physics Laws	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner; areas demonstrating exemplary work may include creating a particularly complex combination of physics laws	Applies physics laws to the circles	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include applying physics laws to more than just one edge of a screen	Does not attempt criterion	20
Brick Collision	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner; areas demonstrating exemplary work may include creating	Alters the state of the bricks upon collision	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include alternating the state of all bricks at an appropriate	Does not attempt criterion	20

	particularly complex brick collision rules such as changing the color or texture over multiple hits		level of complexity		
Circle Collision	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner; areas demonstrating exemplary work may include creating particularly complex rules for circle collision	Alters the state of the circles upon collision	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include altering the state of all circles at an appropriate level of complexity	Does not attempt criterion	20
Code Changes	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner; areas demonstrating exemplary work may include writing a particularly clear explanation that discusses an overall approach to coding strategies	Explains the changes that were made to the code	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include fully explaining all intended changes to the code or adding specific references to the code that was used	Does not attempt criterion	10
Logic	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner; areas demonstrating exemplary work may include creating code that is particularly well modularized and logical	Creates code that follows a logical flow without syntax errors	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include ensuring there are no errors once code is compiled, ensuring the code is executable, or ensuring the flow of the execution reaches all of the code	Does not attempt criterion	5
Best Practices	Exceeds proficiency in an exceptionally clear, insightful,	Applies coding best practices in creations	Shows progress toward proficiency, but with errors or	Does not attempt criterion	5

	sophisticated, or creative manner; areas demonstrating exemplary work may include creating code that shows a strong attention to best practices in both formatting and comments		omissions; areas for improvement may include creating code that is easy to follow and is commented appropriately		
Articulation of Response	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or creative manner	Clearly conveys meaning with correct grammar, sentence structure, and spelling, demonstrating an understanding of audience and purpose	Shows progress toward proficiency, but with errors in grammar, sentence structure, and spelling, negatively impacting readability	Submission has critical errors in grammar, sentence structure, and spelling, preventing understanding of ideas	5
				Total:	100%