Appendices

Appendix A: CS 100 Artifacts

A1: CS 100 – Course Schedule (from Fall 2025 Syllabus)

Week	*Dates*	Topic	Assignment Released	Assignment Due
1	8/25 - 8/31	Python Basics & Operators	8/25 – Mod 1 Guides 8/25 – Lab 1	
2	9/1* - 9/7	Selection Structures	9/1 – Mod 2 Guides 9/1 – Lab 2	9/1 – Mod 1 Guides 9/7 – Lab 1
3	9/8 - 9/14	Graphics: Basic Shapes	9/8 – Mod 3 Guide 9/8 – Lab 3	9/8 – Mod 2 Guides 9/14 – Lab 2
4	9/15 - 9/21	Repetition Structures	9/15 – Mod 4 Guides 9/15 – Lab 4	9/15 – Mod 3 Guide 9/21 – Lab 3
5	9/22 - 9/28	Graphics: Shapes Continued	9/22 – Mod 5 Guide 9/22 – Lab 5	9/22 – Mod 4 Guides 9/28 – Lab 4
6	9/29 - 10/5	10/3 - Exam 1 (Covers weeks 1 – 5)	Practice Exam 1 Released 9/26 Reviewed 10/1	9/29 – Mod 5 Guide 10/5 – Lab 5
7	10/6 - 10/12	Functions and Modularity	10/6 – Mod 6 Guides 10/6 – Lab 6	
8	10/13* - 10/19	Graphics: Event Functions	10/13 – Mod 7 Guide 10/13 – Lab 7	10/13 – Mod 6 Guides 10/19 – Lab 6
9	10/20 - 10/26	Lists & Graphics: Groups	10/20 – Mod 8 Guides 10/20 – Lab 8	10/20 – Mod 7 Guide 10/26 – Lab 7
10	10/27 - 11/2	Strings & File IO	10/27 – Mod 9 Guides 10/27 – Lab 9	10/27 – Mod 8 Guides 11/2 – Lab 8
11	11/3 - 11/9	Dictionaries & Tuples	11/3 – Mod 10 Guides 11/3 – Lab 10	11/3 – Mod 9 Guides 11/9 – Lab 9
12	11/10 - 11/16	11/14 - Exam 2 (Covers weeks 7 – 11)	11/10 – Final Project Practice Exam 2 Released 11/7 Reviewed 11/14	11/10 – Mod 10 Guides 11/16 – Lab 10
13	11/17 - 11/23	Object Classes	11/17 – Mod 11 Guides	
14	11/24* & 12/1 - 12/7	Classes Continued		12/7 – Mod 11 Guides
15	12/8 - 12/11	Finals (No Classes) Please note there is not a final exam		12/8 @ 2:45pm Final Project

* No Classes 9/1→ Labor Day; 10/13→ Mid-Semester Break; 11/25&11/27→ Thanksgiving Recess

A2: CMU-Graphics Interactive Textbook Sample

Note to reviewer: This is an excerpt of interactive textbook chapter I authored for CS 100.

Using onStep for Timed Animations

Unlike `onMousePress`, `onKeyPress`, or `onKeyHold`, which trigger when the user does something, `onStep()` is called repeatedly at a fixed rate; even if the user isn't doing anything! This allows us to create animations or periodically update our app in the background.
First, let's open a testing file.

Open time_events.py in the current window

Example 1: Counting Up

```
from cmu_graphics import *

Label('onStep Demo', 200, 20, size=20, bold=True)
Label('Watch the number increase over time!', 200, 45, size=16)

counter = Label(0, 200, 200, size=100)

def onStep():
    counter.value += 1

cmu_graphics.run()
```

Run this code and watch the `counter` go up continuously. That's `onStep` in action, it's called about 30 times per second by default!

RUN TIME_EVENTS.PY

Example 2: Rotating a Shape

Animation typically means changing some property over time. Here, we rotate a shape:

```
from cmu_graphics import *

Label('Animation Demo', 200, 20, size=20, bold=True)
Label('Watch the star rotate continuously!', 200, 45, size=16)

star = Star(200, 200, 100, 5, fill='gold')

def onStep():
    star.rotateAngle += 1

cmu_graphics.run()
```

Changing `star.rotateAngle` by +1 each time `onStep` is called gives a slow, steady clockwise rotation. Changing to `-=` instead of `+=` makes it rotate counterclockwise.

RUN TIME_EVENTS.PY

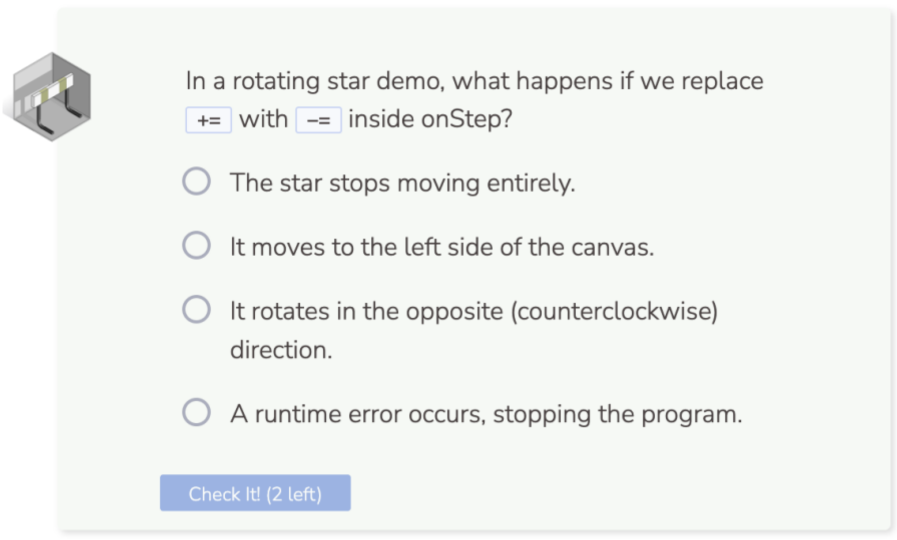
Setting the Speed: app.stepsPerSecond

By default, `onStep` fires 30 times each second. You can increase or decrease this rate like so:

```python
from cmu_graphics import *

Label('Faster or Slower!', 200, 20, size=20, bold=True)
Label('Press f = faster, s = slower, n = normal', 200, 45, size=16)

app.stepsPerSecond = 30
star = Star(200, 200, 100, 5, fill='gold')

def onStep():
    star.rotateAngle += 2

def onKeyPress(key):
    if key == 'f':
        app.stepsPerSecond = 60
    elif key == 's':
        app.stepsPerSecond = 10
    elif key == 'n':
        app.stepsPerSecond = 30

cmu_graphics.run()
```

Here, pressing **f** speeds up the animation, while pressing **s** slows it down. Try it out!

RUN TIME_EVENTS.PY



In a rotating star demo, what happens if we replace `+=` with `-=` inside onStep?

- ○ The star stops moving entirely.
- ○ It moves to the left side of the canvas.
- ○ It rotates in the opposite (counterclockwise) direction.
- ○ A runtime error occurs, stopping the program.

Check It! (2 left)

A3: CMU-Graphics Lab Problem Sample

Note to reviewer: This is an excerpt of a lab problem I authored for CS 100.

Problem 04.04 - Animation with onStep

Open screen_saver.py in current window

The iconic "DVD Logo" screen saver captured the attention of many by bouncing a simple DVD logo around the screen, changing direction each time it collided with a screen edge.

One of the most entertaining aspects was watching and waiting for the logo to perfectly hit a corner! In this project, you will create your own version of this classic screen saver – except instead of "DVD," you'll be bouncing the SUNY Poly logo around.

This problem builds on your knowledge of:

- Shape movement and positioning in CMU Graphics.
- Detecting collisions with screen boundaries.
- Changing properties (like color or direction) when a collision occurs.

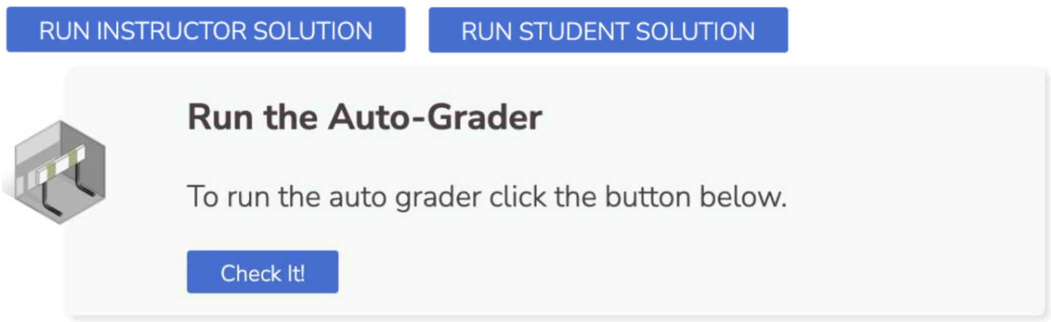
You'll work in the file `screen_saver.py`, which is already set up with some starter code, the SUNY Poly logo, and some initial variables.

Getting Started

Start by running the Instructor Solution (you will need to click into the tab that is connected to the virtual desktop, this will be in the same window as the terminal). Then take a few quick notes on how the logo behaves. Notice how it bounces and what colors it switches to when it hits each boundary. Then, open and edit the student file `screen_saver.py` to replicate that functionality.

Important:

- Do not modify the provided color names or the shapes. The automated tests rely on these!
- Your task is only to implement the bouncing, color-changing logic inside the `onStep()` function.



Run the Auto-Grader

To run the auto grader click the button below.

A4: Final Project Directions and Scoring Rubric

Note to reviewer: This excerpt is the instructions/ scoring rubric of the CS100 Final Project.

Final Project Instructions

Purpose

The final project is your opportunity to showcase your programming skills by designing, developing, and reflecting on a program of your choice. Your project should demonstrate a clear purpose, utilize a variety of concepts learned throughout the course, and reflect thoughtful planning and implementation.

All required deliverables are located within the `starter_files` directory. Your final deliverables must be placed in the `final_submission` directory.

Project Deliverables

1. Design & Development (`starter/Design_Development.docx`)

Program Description: Provide a detailed description of your envisioned program. Clearly state what your application will do. Do not worry about specific implementation details.

Canvas Design: Submit a visual sketch (this can be computer aided or done by hand) of your application's user interface or interaction model. Clearly label any animations or points of user interaction.

Code Design: List and describe at least **eight advanced programming concepts** from this course that you intend to use. Briefly explain the purpose each concept will serve.

2. Implementation (`starter_files/final_project.py`):

Submit your Python file containing your fully implemented project.

- Ensure your code compiles and runs without errors.
- Organize your code neatly (modular is best).
- Use logical naming conventions for variables and functions.
- Include comments to enhance readability and understanding.

3. Reflection (`starter_files/project_reflection`)

Final Code Outline: List and explain at least eight significant coding elements used in your project. For each, include line numbers and describe its contribution to the overall functionality of your program. Highlight the aspect of your project you are most proud of.

Future Development: Reflect on potential improvements or additional features you could add to your project in the future. Describe your ideas clearly and provide high level implementation details.

4. Separation of Duty (`starter_files/team_contributions.docx`)

If working with a partner, detail each team member's specific contributions:

- Clearly outline which functions or code blocks each person developed.
- Describe each member's involvement in the written portions of the project.

Evaluation

Your final project will be graded based on the following rubric .

Category	Strong Evidence	Inconsistent Evidence	Weak or no Evidence	Score
Design & Development - 30%	Student's design of the program clearly demonstrates its purpose & incorporates a wide variety of concepts from the course. **AND** Design is sufficiently ambitious as a showcase of student's current abilities.	Student's design of the program does not demonstrate its purpose or doesn't incorporate a wide variety of concepts. **OR** Design is not sufficiently ambitious as a showcase of student's current abilities.	Student's design of the program does not demonstrate its purpose or doesn't incorporate a wide variety of concepts. **AND** Design is not sufficiently ambitious as a showcase of student's current abilities.	
Implementation - 40%	Code compiled & ran without error. **AND** Code was organized neatly & used logical naming schemes for any variables or functions. **AND** Student used comments in a way that made code more readable & understandable.	Code didn't compile or run without error. **OR** Code wasn't organized neatly or used illogical naming schemes for any variables or functions. **OR** Student hadn't used comments in a way that made code more readable & understandable.	Code didn't compile or run without error. **AND** Code wasn't organized neatly or used illogical naming schemes for any variables or functions. **AND** Student hadn't used comments in a way that made code more readable & understandable.	

Reflection - 30%	Student was able to identify & explain at least eight concepts from the course & how they are applied in their program. **AND** Student described in detail how they may change their implementation, &/or what functionality they would add in future.	Student was not able to identify & explain at least eight concepts from the course & how they are applied in their program. **OR** Student did not describe in detail how they may change their implementation, &/or what functionality they would add in the future.	Student was not able to identify & explain at least eight concepts from the unit & how they are applied in their program. **AND** Student did not describe in detail how they may change their implementation, &/or what functionality they would add in future.	
Bonus - Max 5%	If student(s) demonstrated high quality work in areas including but not limited to originality, cleverness, complexity, engagement, & design bonus may be applied.			
Additional Comments:				
Final Score:				/ 100

Final Submission

Ensure all deliverable files are saved correctly in the `final_submission` directory.

Good luck!