

## MISSION 15: Handball

Time: 60-120 minutes

### Overview:

Ready to build a truly iconic video game? This is the first of a two-part mission sequence. This mission will lead students on a step-by-step journey to develop a retro video game of *American Handball*. The game is like a 1-player version of the classic “Pong”. The program is continued in Mission 16.

### Cross Curricular:

- **MATH:** The mission includes a lot of advanced math, especially trigonometry. Review angles or how to convert from radians to degrees.
- **SCIENCE:** The animation of the ball and the paddle uses physics, specifically velocity and distance traveled. Use the game as the introduction or ending to a physics unit.
- Supports **language arts** through reflection writing.

### Materials Included in the learning portal [Teacher Resources](#):

#### Mission 15 Assignment

The assignment is the worksheet for students to complete as they work through the mission. It should be given digitally to each student before the mission starts. They answer questions in the document during the mission and turn it in at the completion of the mission.

#### Mission 15 Lesson Plan

The lesson plan comes from the original CodeX Teacher Manual and is included here for easy reference.

#### [Mission 15 Solutions](#) (Handball)

### Formative Assessment Ideas:

- Exit ticket(s)
- Quizzes in CodeSpace
- Assignment document completion
- Completed **Handball** program
- Daily student reflections
- No specific reviews are prepared for this mission. New concepts are not introduced but are applied. You can use the previous review Kahoots as needed.

### Vocabulary:

- **Physics engine:** A device that uses the mechanics of velocity, distance and time
- **Initialization (review):** Set the initial or first value of a global variable when the program starts
- **Delta time:** Elapsed time in milliseconds (or change in time)
- **UX:** User experience

### Preparing for the lesson:

This mission will create a one-player handball game, similar to the old-fashioned “pong.” Some objectives are heavy in math, especially trigonometry. Some earlier objectives also go into detail about physics with velocity, distance and time. If the math is too advanced for your students, they can still complete the mission using the code that is given and reading through the hints. They are not required to understand all the math

equations and concepts.

Also, students will create a lot of functions and include many global variables and constants. It is a lot to keep track of, so students will need to pay attention to details and type carefully. They should scroll through the code in CodeTrek to make sure they are placing new code in the correct functions and with the correct indenting. Make sure each objective is correct and code works properly before going to the next objective. Otherwise it will be very difficult to debug.

Students will use the Codex throughout the lesson. Decide if they will work in pairs or individually.

- There is no slide deck or workbook for this mission. All instructions are included in CodeSpace and CodeTrek.
- Be familiar with the Assignment document and the questions they will answer.
- Prepare the Assignment document for each student digitally using your LMS.
- The mission program does not need to be portable. If you want students to use the CodeX without a cable, then have batteries available.

## Lesson Tips and Tricks:

### Pre-Mission Discussion:

A pre-mission prompt is not given for this lesson. Students will apply what they have been learning about Python code in this two-part program. After reviewing the lesson, you may come up with your own pre-mission discussion prompts, or you may want to just jump in and start the mission.

### Mission Activities:

Most of this lesson is on the computer, writing code to create a single player game.

- Each student will complete an Assignment Document.
- Students could work in pairs through the lesson, or can work individually.
- Students will need the CodeX and USB cable.

#### Teaching tip: Mission Introduction


This objective introduces the mission by showing the two parts to the completed handheld game and the game plan for this mission. The completed program will be a one-player version of “pong”.

#### Teaching tip: Objective #1

This objective includes two functions: one will draw the ball and one will serve the ball. It uses a bit of math for the location and speed of the ball and the time it takes to move across the screen. It uses a delta for time, similar to the deltas used in Mission 14.

The ball will go off the screen.

#### Teaching tip: Quiz


Students take a  short quiz. The 3 Quiz questions are below.



 **Teaching tip: Objective #2**

Students will modify their code to make the ball bounce off the walls instead of disappearing.

Students will answer a question on the assignment document.

 **NOTE:** Students should use the HINT to help answer the question, and to complete the code.

 **Teaching tip: Objective #3**

Students will add another function to return the elapsed time. The HINT is useful.

Students will answer a question on the assignment document.

 **NOTE:** Not many lines of code, but each line is important, so be careful not to skip a step.


 **Teaching tip: Quiz**

Students take a **?** short quiz. The 3 Quiz questions are below. They are all about the new elapsed time function.

 **Teaching tip: Objective #4**

Students use a tuple to keep track of both the X and Y position, and a list to keep track of the two velocities. Why not both be one or the other? A list is changeable – you can do math with it, and a tuple is not.

Students will answer a question on the assignment document. (Compare a tuple to a list.)


 **NOTE:** There is a LOT going on in this objective. Pay attention to details!

 **Teaching tip: Objective #5**

Students will add a function for the screen layout. It needs to be called once at the beginning of the main program. Also, the if statements for collision need to be adjusted.

 **Teaching tip: Objective #6**


Students add sound to the game in the form of beeps. A new function is added and called in the collisions. Several variables and constants are added. Again – pay attention to details!

 **NOTE:** Remember to “from soundlib import \*”

 **Teaching tip: Objective #7**

Define two more functions, one to draw the paddle and one to check the button presses. Then call the functions in the correct places. With so much going on, it is really important to debug as you go. If a student sees that something is off, fix it before going on!!

Students will answer a question on the assignment document.

 **NOTE:** The “sound\_cut” variable is moved from below the tone constants to above. It doesn’t matter either way, so don’t let this trip up your students.



 **Teaching tip: Quiz**

Students take a **?** short quiz. The 3 Quiz questions are below.

 **Teaching tip: Objective #8**

The paddle collisions are explained and coded. Careful coding here!

 **Teaching tip: Objective #9**

MMake the project into a game by adding game elements - ball can go off the bottom of the screen, re-serve the ball, etc.

Students will answer a question on the assignment document.

 **Teaching tip: Objective #10**

Make the project into a game by adding a score.

Students will answer a question on the assignment document.

 **Teaching tip: Objective 11**

A lot of little details here, in different functions. Not complicated, but students need to pay attention to details.


Students will answer a question on the assignment document.


 **Teaching tip: Quiz**

Students take a **?** short quiz. The 1 Quiz question is below. It is kind of a tricky one using “continue”. You may want to trace this one with your students.

 **Teaching tip: Objective #12**

This objective uses angles for the bounce. A new function is created for the hit ball angle. More math (a little trig)!

 **NOTE:** The global variable “ball\_speed” needs to be defined near the top of the program under “serve\_timer”. This may not be explicitly said in CodeTrek, so be aware if students get an error message about the variable undefined.

 **NOTE:** The second HINT mentions a line of code to add to the program. It probably isn’t necessary to do this, but it doesn’t hurt to do it, either.

 **Mission Complete:**

This mission ends with a completed, working program that will play a one-player handball game similar to pong. You need to decide how you will use the program for assessment. You could:



- Go to each student and check-off their code
- Have the students download their code to a text file and turn it in using your LMS or as an attachment in email
- Have students print their code (either download and then print the text file, or print a screenshot)
- Have students switch computers and run each other's code. Fill out a simple rubric and turn in to teacher
- Any other way that works for you

### Post-Mission Reflection:

The assignment document does not include a post-mission reflection. You can add one if you want to.

End by collecting the Assignment document and any formative assessment you want to include.

### IMPORTANT Clearing the CodeX:

The students have already created a “Clear” program. Students should open and run “Clear” at the end of each class period.

### SUCCESS CRITERIA:

- Define and call 12 functions.
- Draw a screen layout for the game.
- Use CodeX buttons as event handlers for the game.
- Keep track of score and lives.
- Create 2 animated objects in the game: paddle and ball.
- Game works correctly, showing a ball bounce, paddle movement, and score

## ? Quiz Questions

### Quiz 1



Start Up!

How many *milliseconds* are in 1 second? +5 XP

0.001  1000  100  1 million

Say the ball moves at a velocity of  $\frac{1}{2}$  pixel per millisecond. +5 XP

- How far would it move in 10 milliseconds?

50 pixels  10 pixels  5 pixels  20 pixels

Your *game loop* uses a *global* variable `ball_v` for the ball's velocity. Where is this *variable* initialized? (*first assigned to*) +5 XP

At the beginning of the game loop.  Inside the `draw_ball()` function.

Inside the `serve_ball()` function.

## Quiz 2

**Delta Force**

What do the letters D T stand for in the `dt` variable? **+5 XP**

"dog tired"  "delta time"  "difference time"  "data test"

"delta tricep"

What would you expect the value of `dt` to be after the following code runs? **+5 XP**

```
elapsed_ms()
time.sleep_ms(42)
dt = elapsed_ms()
```

Error: No target for assignment.  10  40  42

How does the `elapsed_ms()` function "remember" the millisecond value the last time it was called? **+5 XP**

The `ms` global saves milliseconds between function calls.

Computers don't remember. Eschew anthropomorphism.

All `variables` inside a function are preserved across calls.

## Quiz 3

**Midway!**

Which *three* of the following comparisons are `True`?

`(1, 2, 3) == (3, 2, 1)`  `1 < 0`  `("Right", "On") == ("Right", "On")`

`(1, 2, 3) == (1, 2, 3)`  `10 > 9`  `"one" == 1`

What's the purpose of the `sound_cut` variable in your Handball program?

To count up seconds until sound stops.  It is the cut-off frequency of the sound.

To count down the milliseconds till you turn off the sound.

What is `max(min(3, 2), 1)`

3  4  1  2

## Quiz 4

What is the value of `count` after the following code runs?

```
count = 0
x = 0
while x < 5:
    x = x + 1
    if x == 2:
        continue
    count = count + 1
```

 3 5 4