# UCF Cardiac 2.0 Application

© 2023 University of Central Florida. All rights reserved.

**Class: Case Manager**

* DeselectAllToggles()
  + Deselects all toggle buttons.
* GetRequest()
  + Uses UnityWebRequest to get cases information
    - If Webrequest fails, or uri is incorrect
      * Calls **m2Cases()** using the data from CaseQuestion class
    - Else
      * Grabs the response from PHP app as JSON
      * Formats the data
      * For each cases prompt
        + Adds deleted ‘{‘ and removes trailing ’,’
        + Removes last ‘]’
* SetAnswerImages()
  + For each answer display, set the image to the image for the current case question
* Start()
  + Deactivate the result prompt
  + Read in the case questions
  + Count the number of M1 cases
  + Set numberofM2Questions to the total number of cases minus the total number of M1 cases
  + Run StartRoutine function that calls **GetRequest()** which gets the case questions
* Submit()
  + Deactivate the submit button
  + Stops the timer for the case
  + Saves the time spent on the case in the “hh:mm:ss” format
  + Restart the timer for the next case
  + Saves the selected Answer choice for the case
    - If the previouscase index does not equal -1 then sets the previous case button active
    - Else, deactivates the previous button
    - If the resultSelection is the correct answer
      * Sets the resultPrompt and correctPrompt to active
      * Sets the incorrectPromts to inactive
      * Outputs “Correct” on the screen
      * Assigns the grade to the case
    - Else
      * Sets the resultPrompt and incorrectPrompt to active
      * Sets the correctPromt to inactive
      * Outputs “Incorrect” to the Screen
      * Assignes the grade to the case
  + Calls StartCoroutine function that calls the **Upload()** function
* Upload()
  + Creates a WWW Form with the case data
  + POSTS a UnityWebRequest to the database with case attempt
  + If there is a connection error:
    - Logs error
  + Else if there is another error:
    - Logs the error response from the PHP app
  + Else
    - Returns the log with the SID
* atrialFibr()
  + Sets the current active heart model to atrialfib and deactivate all others
  + Calls **resetHeartPosition()**
* atrialFlut()
  + Sets the current active heart model to atrialflutter and deactivate all others
  + Calls **resetHeartPosition()**
* avnrt()
  + Sets the current active heart model to avnrt and deactivate all others
  + Calls **resetHeartPosition()**
* getAnswer()
  + If the first answer is selected then sets the result selection to A
  + Else If the second answer is selected then sets the result selection to B
  + Else If the third answer is selected then sets the result selection to C
* getCase()
  + Gets the current case index, text, result description, and the correct answer to the case
  + If the current case index is 0, then sets the case title to “Case 1” and call **atrialFibr()**
  + Else If the current case index is 1, then sets the case title to “Case 2” and calls **atrialFlutt()**
  + Else If the current case index is 2, then set the case title to “Case 3” and calls **avnrt()**
  + Starts the timer to case question
* m2Cases()
  + For the total number of M2 cases, creates case question objects
* nextCase()
  + Sets the previousCaseIndex to the currentCaseIndex
  + Stops the timer
  + Saves the time spent on the case attempt
  + If the currentCaseIndex is less than 2
    - Increases the currentCaseIndex by 1
    - Calls **getCase()** for the next case
    - Deactivates the resultPrompt
    - Activates the submit button
    - Calls **DeselectAllToggles()**
  + Else
    - Resets the currentCaseIndex to 0
    - Calls **getCase()**
    - Deactivates the resultPrompt
    - Activates SubmitButtom
    - Calls **DeselectAllToggles()**
  + Resets the stopwatch
* previousCase()
  + Stops the timer and record the time spent
  + Sets the currentCaseIndex to the previousCaseIndex
  + Sets the previousCaseIndex to -1
  + Calls **getCase()** with the currentCaseIndex
  + Deactivates the ResultPrompt
  + Activates the submit button
  + Calls **DeselectAllToggles()**
* resetHeartPosition()
  + Resets the heart position, rotation, and its scale.
* sinus()
  + Set the current active heart model to sinus and deactivate all others
  + Reset the heart position
* tryAgain()
  + Deactivates the result prompt
  + Sets the submit button to active
  + Calls **DeselectAllToggles()**

**Class: CaseQuestion**

* CaseQuestion()
  + Constructor to create a case question.
* CreateFromJSON()
  + Creates CaseQuestion object from JSON string

**Class: ChangeScene**

* LoadScene()
  + Loads the scene from string passed in by the inspector.

**Class: CountdownTimer**

* Start()
  + Sets the current time to the starting time
* Update()
  + Calculate the current time
  + If the current time is less than or equal to 0, then set the current time to 0

**Class: CSVReader**

* Read()
  + Loads the resources from the .CSV
  + Splits the data to lines
  + Splits the header from the lines
  + For each line
    - Split line into values
    - Creates new Dictionary entry
    - For each column in CSV
      * Formats the values
      * Updates entry with appropriate values
    - Updates the list with values
  + Returns list with values read from CSV

**Class: DataManager**

* Awake()
  + If Instance is not null
    - Destroys the gameObject
  + If scene selected is “LoginScene” and SID is “”
    - Starts coroutine **GetRequest()**
  + Sets instance to this
  + Calls DontDestroyOnLoad() on gameObject
* GetRequest()
  + Creates a UnityWebRequest
  + If Request returns a Connection error
    - Logs the connection error
  + Else if the PHP response equals to error2 (no students found in the database
    - Logs the error
  + Else
    - Logs the PHP response
    - Stores the data returned from PHP script

**Class: Dialogue**

* Dialogue()
  + Constructor to create a Dialogue object

**Class: FlashcardManager**

* Start()
  + Reads cardiac\_flashcards.csv
  + Deactivates correct and incorrect buttons
  + Gets DataManager object (stores student information) and sets it to savedData for later use
  + Calls GetRequest() inside a StartCoroutine method
* Update()
  + Controls flip animation
* NextCard()
  + Stops the watch, gets elapsed time and formats string to “hh:mm:ss”
  + Sets time, for specified Question instance, to formatted elapsed time string
  + Sets grade, for specified Question instance, to passed grade value
  + Deactivates correct and incorrect buttons
  + Resets slider position back to the center
  + Calls Upload() inside a StartCoroutine method
  + Increases cardNum
    - If cardNum value exceeds the number of cards sets value back to 0
  + Sets confidence, for specified Question instance, to the middle value
  + Changes cardCounter text to reflect progress through the flashcards
  + Creates new instance of StopWatch and starts the timer
* getConfidence()
  + Sets confidence, for current Question instance, to the passed confidence value
* FlipCard()
  + Activates correct and incorrect buttons
  + Sets timeCount, isFlipping, and isShrinking to appropriate values to be used for flip animation
* m1Cards()
  + Sets Prompt and Answer, for each instance of Question, from cardiac\_flashcards.csv file using only M1 questions and answers
* m2Cards()
  + Sets Prompt and Answer, for each instance of Question, from cardiac\_flashcards.csv file using only M2 questions and answers
* GetRequest()
  + Gets current scene name
  + Sets val to 1 if current sceneName is M1HeartScene, else sets it to 2
  + Sets uri to our databases GET url with val appended at the end (val being the section number)
  + Creates a web request, using the uri, and waits for the page
  + If web request fails
    - Get flashcard data locally, from csv
  + Else
    - Formats JSON so can be used to set Question class members
  + Sets default confidence
  + Displays flashcard question and flashcard counter
  + Starts timer
* Upload()
  + Creates form with fields matching that of database
  + Creates web request using the POST url for our database, waits for page
  + Checks if web request was successful and uploaded data

**Class: IntroToScene**

* Start()
  + Deactivates the scene content
* showCasesContent()
  + Calls **showContent()** to display the case content
  + Finds the case question based on the case number
* showContent()
  + Sets active the scene content
  + Deactivates the intro prompt

**Class: LoadSceneAfterTime**

* Update()
  + Measures the time elapsed
  + If time elapsed is greater than 20 frames
    - Calls **LoadScene()** to load a new specified scene

Class: Login

* Login()
  + Creates a Login object from JSON

**Class: LoginValidation**

* Awake()
  + Sets the errorText.enabled to false
  + Sets the errorText.object to false
* CheckValidation()
  + Starts Coroutine **GetRequest()**
  + Checks if the login credentials are valid
  + If either of the login credentials are empty
    - Sets the errorText to true
* GetRequest(string uri)
  + Used to satudents
  + Creates a UnityWebRequest for PHP app
  + If webRequest returns a connection error
    - Sets error text to "No connection detected, use Guest Login"
    - Enables error text on the screen
  + If webRequest return invalid password
    - Logs the error on the screen of the user
  + If webRequest returns no students found in the database
    - Logs error “Access Code not recognized”
  + Else
    - Login successful
    - Calls **CreateFromJSON()** to save user data
    - Saves the user data in the savedData object
    - Loads the next Scene
* GetRequest(string uri, string aInput, string pInput)
  + Used to login guest users
  + Creates a UnitryWebRequest for PHP app
  + If webRequest returns a connection error
    - Logs error
    - Sets object savedData to Offline Guest
    - Loads the next scene
  + Else
    - Online Login successful
    - Calls **CreateFromJSON()** to save user data
    - Saves the user data in the savedData object
    - Loads the next Scene
* Start()
  + Set the first canvas to active
* guestAccess()
  + Calls a coroutine **GetRequest()** using the username and password credentials as “guest” and “guest” for Guest users

**Class: M1HeartControls**

* Start()
  + Sets the current time to the start time
  + Deactivates the heart anatomy
* Update()
  + Updates the current time
  + If the current time is less than or equal to 0 and startReset is true
    - Sets currentTime to 0
    - Sets startReset to false
    - Call **resetHeartPosition()**
* anatomy()
  + Toggles to activate and deactivate the heart’s anatomy
* bradycardia()
  + Decreases the animation speed and the ekg speed to 0.5
* normal()
  + Sets the animation speed and the ekg speed to 1
* resetHeartPosition()
  + Resets the heart model position and scale
  + If the speedState is 0, calls **bradycardia()**
  + Else if the speedState is 1 calls **normal()**
  + Else if the speedState is 2 calls **tachycardia()**
* tachycardia()
  + Increases the animation speed and the ekg speed to 2
* tutorial()
  + Toggles to activate and deactivate the tutorial warning

**Class: M2HeartControls**

* Start()
  + Sets the current time to the start time
  + Sets the anatomy of the heart to false
  + Sets the blood and electrical flows of the heart to false
* Update()
  + Updates the current time
  + If the current time is less than or equal to 0 and startReset is true
    - Sets currentTime to 0
    - Sets startReset to false
    - Call **resetHeartPosition()**
* anatomy()
  + Toggles to activate and deactivate the heart’s anatomy
* atrialFibr()
  + Sets the current heart model to atrial fibrillation
  + Activates the EKGs and heart model for atrial fibrillation
  + Deactivates all the other cardiac symptom models and EKGs
  + If the speedState is 0
    - Calls **slow()**
  + Else if the speedState is 1
    - Calls **normal()**
  + Else if the speedState is 2
    - Calls **fast()**
  + Sets the title and text for the prompt for the symptoms
* atrialFlut()
  + Sets the current heart model to atrial flutter
  + Activates the EKG for atrial flutter
  + Deactivates all the other cardiac symptom models and EKGs
  + If the speedState is 0
    - Calls **slow()**
  + Else if the speedState is 1
    - Call **normal()**
  + Else if the speedState is 2
    - Calls **fast()**
  + Sets the title and text for the prompt for the symptoms
* avnrt()
  + Sets the current heart model to avnrt
  + Activates the EKG for AVNRT
  + Deactivates all the other cardiac symptom models and EKGs
  + If the speedState is 0
    - Calls **slow()**
  + Else if the speedState is 1
    - Calls **normal()**
  + Else if the speedState is 2
    - Calls **fast()**
  + Sets the title and text for the prompt for the symptoms
* blood()
  + If the heart model is not transparent
    - Sets model to transparent
    - Sets isBloodTransparent to true
  + Else
    - Sets isBloodTransparent to false
  + If both blood and electrical are not transparent
    - Sets the material back to its original
  + Toggles the blood animations for the cardiac symptoms
* electric()
  + If the heart model is not transparent
    - Sets it to transparent
    - Sets isElectricTransparent to true
  + Else
    - Sets isElectricTransparent to false
  + If both blood and electrical are not transparent
    - Sets the material back to its original
  + Toggles the electric animations for the cardiac symptoms
* fast()
  + Sets speedState to 2
  + Sets all animation speeds to 1.5
  + Sets the animation speed for the blood and electrical particles to 1.5
* normal()
  + Sets speedState to 1
  + Sets all animation speeds to 1
  + Sets the animation speeds for the blood and electrical particles to 1
* resetHeartPosition()
  + If the heart model is sliced
    - Resets its location and scale of sliced model
  + Else,
    - Resets the position and scale of the normal heart model,
    - If the heart was transparent before switching
      * Sets heart material to transparent
    - Else
      * Sets heart material to original
  + If the speedState is 0
    - Calls **slow()**
  + Else if the speedState is 1
    - Calls **normal()**
  + Else if the speedState is 2
    - Calls **fast()**
* setHeartActive()
  + If the active heart model does not equal to null
    - Sets it to active
* setHeartInactive()
  + If the active heart model does not equal to null
    - Sets it to inactive
* sinus()
  + Sets the current heart model to sinus
  + Activates the EKG for sinus
  + Deactivates all the other cardiac symptom models and EKGs
  + If the speedState is 0
    - Calls **slow()**
  + Else if the speedState is 1
    - Calls **normal()**
  + Else if the speedState is 2
    - Calls **fast()**
  + Sets the title and text for the prompt
* slice()
  + If the heart is not sliced
    - Sets isSliced to true
    - Activates the slice heart model
    - Deactivates the normal heart model
    - For each feature in upcomingSlicedHeartFeatures
      * Sets the feature to active
    - For eact text in comingSoonText
      * Actiaves the object to coming soon text for the sliced heart model
  + Else
    - Deactivates the sliced heart model
    - Activates the normal heart model
    - For each feature in upcomingSlicedHeartFeatures
      * Activates the feature
    - For each text in comingSoonText
      * Deactivates the gameObject
  + Calls **resetHeartPosition()**
* slow()
  + Sets speedState to 0
  + Sets animation speeds to 0.5
  + Sets the animation speed for the blood and electrical particles to 0.5
* tutorial()
  + Toggles to activate and deactivate the tutorial warning

**Class: NavBarManager**

* Start()
  + Sets flashcardSection to false
  + Sets caseSection to false
  + Calls **StartPrompt()**
* StartPrompt()
  + Sets the current scene
  + If the current scene is M1
    - Sets the text for the title and the prompt
* afterload()
  + Activates heartSection and flashcardSection
  + Deactivates casesSection and casesObject
  + Activates heartModels
  + Calls **resetHeartPosition()**
  + Sets the prompt title and text
* antrioventricularNode()
  + Sets the title and text for the antrioventricularNode prompt
* aorta()
  + Sets the title and text for the aorta prompt
* cases()
  + Deactivates heartSection and flashcardSection
  + Activates casesSection and casesObjects
  + Deactivates heartModels
* contractility()
  + Activates heartSection
  + Deactivates the caseSection, flashcardSection, casesSection, and casesObject
  + Activates heartModels
  + Calls **resetHeartPosition()**
  + Sets the text from the prompt title and the prompt text
* flashcards()
  + Activates flashcardSection
  + Deactivates heartSection, casesSection, and casesObject
  + Activates heartModels
* leftAtrium()
  + Sets the title and text for the leftAtrium prompt
* leftVentricle()
  + Sets the title and text for the leftVentricle prompt
* preload()
  + Activates heartSection
  + Deactivates flashcardSections, casesSection, and casesObject
  + Activates heartModels
  + Calls **resetHeartPosition()**
  + Set the prompt title and text
* rhythms()
  + Activates heartSection
  + Deactivates flashcardSections, casesSection, and casesObject
  + Activates heartModels
  + Calls **resetHeartPosition()**
* rightAtrium()
  + Sets the title and text for the rightAtrium prompt
* rightVentricle()
  + Sets the title and text for the rightVentricle prompt
* sinoatrialNode()
  + Sets the title and text for the sinoatrialNode prompt

**Class: Question**

* Class to hold all flashcard data being sent to database
* Question()
  + Constructor that sets prompt (aka the question) and answer
* CreateFromJSON()
  + Sets the values of Question class members that are contained in JSON string

**Class: SliderController**

* HasSliderChanged()
  + Checks if the slider value has changed by finding the difference between the slider value and the initial value
* ResetSliderValue()
  + Resets the value of the slider to the default
* Start()
  + Sets the initial value of the slider

**Class: SliderEventHandler**

* OnInteractionEnded()
  + Sets slider value
  + Calls getConfidence from FlashCardManager

**Class: TutorialManager**

* DisplayMessage()
  + Displays the appropriate Tutorial Message on the screen
* NextMessage()
  + Updates the Tutorial Message
  + Checks what Message is selected
  + If Message 3 (Moving object) is selected
    - Shows the heart model
    - Hides “Continue” button
    - Calls **resetHeartPosition()**
    - Caches the heart model current position
  + If Message 5 (Rotating object) is selected
    - Removes the Continue button
    - Caches the heart rotation
  + If Message 7 (Expanding) is selected
    - Removes Continue button
    - Caches the heart model’s current size
  + If Message 9 (Scaling down) is selected
    - Removes Continue button
  + If Message 11 (Slider) is selected
    - Removes the Heart model from the screen
    - Shows the Slider on the screen
    - Removes continue button
  + If Message 13 (Final Message) is selected
    - Removes the Sliderand Continue button from the screen
  + If active Message value is smaller than the length of the currentMessages
    - Dislays the tutorial prompt on the screen
  + Else
    - Prints the debug line that tutorial has ended
* TutorialMessages(**Dialogue** [] )
  + Stores an array of prompt messages for the tutorial
* resetHeartPosition()
  + Resets the heart to the default position
* Start()
  + Called before the first frame update
  + Calls **tutorialMessages()**
  + Hides heart model and slider from the screen
* Update()
  + Called once per frame
  + Checks which active message is selected
  + If Message 3 (Moving Object) is selected
    - Gets the current position of the Heart model
    - If user moved the object
      * Shows Continue button
      * Calls **NextMessage()**
  + If Message 5 (Rotating) is selected
    - Gets the current rotation of the Heart model
    - If user rotated the object
      * Shows Continue button
      * Calls **NextMessage()**
  + If Message 7 (Expaning) is selected
    - Gets the current size of the Heart model
    - If user enlarged the object
      * Shows Continue button
      * Calls **NextMessage()**
  + If Message 9 (Scaling down) is selected
    - Gets the current size of the Heart model
    - If user scaled down the object
      * Shows Continue button
      * Calls **NextMessage()**
  + If Message 11 (Slider) is selected
    - Checks if Slider position has Changed
      * Shows Continue button
      * Calls **NextMessage()**
  + If Message 13 is selected
    - Sets a timer
    - Once time is greater than the delay time, loads a new Scene