* **PlayState.hx**
  + Initializes the cameras used
    - utilities/camera/Camera.hx is a wrapper class for FlxCamera.
  + Create background (FlxSprite)
  + Create teams
    - team/Team.hx
  + Create players and give them a controller
    - player/Player.hx
    - utilities/controller/Controller.hx
  + Each player is given a battle menu. Each menu is assigned a position on screen.
    - player/menu/BattleMenu.hx
  + Create player characters. Each character is a super class of the player character class.
    - player/PlayerCharacter.hx
    - player/playerCharacterClass/Wizard.hx
    - player/playerCharacterClass/Paladin.hx
    - player/playerCharacterClass/ Fighter.hx
    - player/playerCharacterClass/ Ranger.hx
  + Create monster group RhinoGroup.hx
    - monsters/MonsterGroup.hx
    - monsters/monsterGroup/RhinoGroup.hx
* **library/Library.hx** – Enum of classes used by all elements of the project
  + library/Players.hx
    - Tracks all players
    - Each player created is given a unique ID (0-99)
  + library/Controllers.hx
    - Tracks all controllers
  + library/Character.hx
    - Tracks all characters
    - Give each character a unique ID (1000-9999)
    - Can retrieve arrays of other active characters by team:
      * self
      * all
      * allies
      * alliesExcludingSelf
      * others
      * enemies
        + Each function requires the caller to supply their self as an argument
        + All characters are enemies of characters whose team is null
        + Each of the above functions has an optional argument to include dead characters
      * deadAllies
      * deadOthers
      * deadEnemies
        + Each of these has a placeholder argument to ensure they have the same signature as the normal functions
        + Currently monster characters are instantly deleted when they die
  + library/Teams.hx
    - Track all teams
    - Give each team a unique id (100-999
  + library/CardOverseer.hx
    - Track all cards by owner’s character ID
    - Give each card a unique ID (10000-infinity)
  + library/UniqueIDBot.hx
    - For all other objects that need a unique ID
    - Gives ID’s from -1 to negative infinity
  + library/Cameras.hx
    - Enum of cameras used by game
  + library/ElementalResistances.hx
    - Enum of damage resistances/vulnerabilities by element type
    - Get function (takes DamageTypes, returns Resistances)
  + library/StatTables.hx
    - Table of ability modifiers by stat level
  + library/DamageColors.hx
    - Lookup for color by DamageTypes
  + library/StatusColors.hx
    - Lookup for color by StatusTypes
  + library/CardColors.hx
    - Lookup for color by CardFamily
* **utilities/plusInterface**
  + An interface that extends functionality built into FlxSprite, FlxBasic, and FlxState
  + Each plus object has a parent and tracks sub objects
  + Destroying a plus object also destroys all sub plus objects
  + Provides an alternate update method (plusUpdate) which can allows objects to pause updating of an object and all of its sub objects
    - These object’s normal update function is still called
    - This is useful for delay/freeze/stop status effects and more
    - plusUpdate updates the leaf objects first and root last
    - objects have a bool forceUpdate that overrides this
  + Each plus interface object requires a parent object at creation
    - The parent tracks the new object
    - The newly created object tracks its parent
  + PlusInterface.hx - Interface object
    - private var plusType:PlusEnum;
    - public function type():PlusEnum;
    - private var trackedItems:Array<PlusInterface>;
    - private var tracker:PlusInterface;
    - public function addItem(item:PlusInterface):Void;
    - public function setTracker(tracker:Null<PlusInterface> = null):Void;
    - public function removeItem(item:PlusInterface):Bool;
    - private var updateEnabled:Bool;
    - public var forceUpdate:Bool;
    - public function setUpdate(enabled:Bool):Void;
    - private function plusUpdateSubroutine(elapsed:Float):Void;
    - public function plusUpdate(elapsed:Float):Void;
    - public function destroy():Void;
  + PlusEnum.hx
    - Enum of available plus interface objects
    - Used to by plusType to determine object’s type
  + BasicPlus.hx
    - extends FlxBasic and implements PlusInterface
  + BasicPlusTracker.hx
    - implements PlusInterface and intended to be used much like an array of BasicPlus objects
  + SpritePlus.hx
    - extends FlxSprite and implements PlusInterface
  + SpritePlusTracker.hx
    - implements PlusInterface and intended to be used much like an array of SpritePlus objects
  + StatePlus.hx
    - extends FlxState and implements PlusInterface
    - StatePlus does not take a parent object on creation
      * Intended to act as the root for objects in the state
      * Has the ability to disable all plusUpdate calls for all objects in state
        + includes objects where forceUpdate=true
  + TimerPlus
    - Extends BasicPlus
    - A timer that calls a TimerPlus->Void function when the timer ends
      * Uses itself as an argument for the TimerPlus argument
      * Destroy itself when call completed
    - Makes the function call first if it would be destroyed
* **utilities**
  + utilities/animation/Animation.hx – wrapper for FlxSprite.Animation
    - extends SpritePlus
    - Takes arguments for the number of frames in the animation and the runtime of the animation
    - Optional argument for the number of times the animation loops (will play for a total of loops\*runtime seconds)
    - play and stop functions
  + utilities/button/Button.hx
    - Extends SpritePlus
    - Designed to be used with a mouse. Many objects extend this class but mouse functionality isn’t being used
  + utilities/camera/Camera.hx
    - Extends FlxBasic. Wrapper class for FlxCamera
    - Mostly a holdover from old Far Horizons project. Mostly functions like a FlxCamera
    - function flxCam returns FlxCamera object
  + utilities/controller/Controller.hx
    - Acquires the first available active and unclaimed FlxGamepad
    - Each button has an integer tracking how long it’s been held down or how long it’s been released in number of frames
      * 1 = button just pressed
      * >2 = button held for 2 or more frames
      * 0 = button just released
      * <-1 = button released for 2 or more frames
  + utilities/event
    - Classes and enums used for mouse events (not used)
  + utilities/observer
    - Observer and subject classes. Originally used with mouse interface in Far Horizons (not used now)
  + utilities/Selection
    - (TODO move to player folder?)
    - A selection of buttons
    - Used to track possible options that the player (Player.hx) can choose from
    - Player class provides controller inputs
    - Tracks Void->Void functions that will be called when a given button is pressed
      * Generally these functions iterate through the selection and activate functions like playing a card
    - Player class has a base selection. This selection can have sub selections. When calling a function in a selection, the super selection will pass the call down to the lowest sub selection.
      * This makes selections act like stacks
        + Selections can be pushed onto the stack and popped off
    - Can have multiple “slices”. Generally this is used when playing a card that primarily targets enemies but can target allies (or vice versa)
* **player**
  + Player.hx
    - Extends SpritePlus
    - Visually represented by a cursor
    - Moves between various buttons (its target)
    - Has a BattleMenu, Selection, PlayerCharacter, and Controller
    - Its base Selection is provided by the BattleMenu
  + player/menu
    - BattleMenu.hx
      * Contains MenuButtons, Selection, healthBar, staminaBar, manaBar, and guardText
      * Contains functions for interacting with BattleMenu through its selection
      * Updates healthBar, staminaBar, manaBar, and guardText with values from its owner’s (player’s) character
    - MenuButton.hx
      * Refrences a card in the player’s hand
      * Text and color change based on card
  + PlayerCharacter.hx
    - Extends Character
    - Function for adding card to deck
    - PlayerCharacters draw and discard cards differently than MonsterCharacters
      * When a player discards a card the card slot is replaced with a temporary unplayable card called Redraw. Only players use Redraw
      * One a Redraw charges it has the player remove it and replace it with a card from the player’s deck. (This uses the function drawPlayerCard)
      * If a card in the player’s hand is null, then the player will replace the card with a Redraw
      * If the player has too many cards in hand then extra cards are discarded (function removeCardSlot)
    - When a player plays a card, it resets all cards in hand that match the played card’s CardFamily
    - Ability to kill and revive PlayerCharacters
    - override updateStats: update the menu when stats change (change number of buttons in selection when hand size changes)