Design Overview for Collecticopter

Summary of Program

Collecticopter is a multi-level game in which the player flies around collecting items that randomly appear on screen, then uses them to 'tame' the creatures present in each level. The three levels of the game are shown in figures 1, 2, and 3.

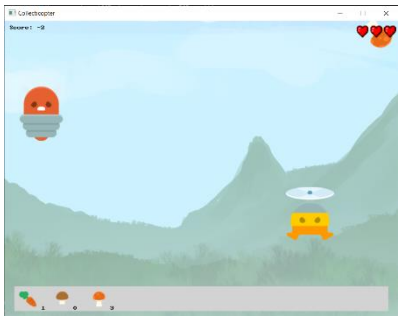


Figure 1. Level 1

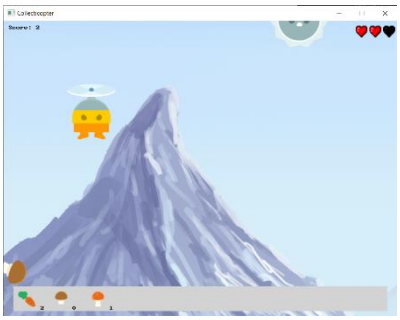


Figure 2. Level 2

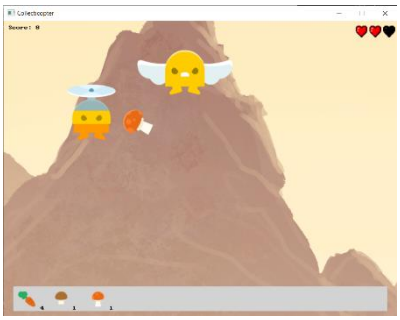


Figure 3. Level 3

Collecting items either increases or decreases the player's score depending on the type of item. Taming each creature requires the player to have a certain number of points, and the player must have a certain number of the item that the type of creature you are trying to 'tame' wants. The relevant items are taken out of the player's inventory when the creature is 'tamed'. If the player collides with the creatures before having enough points and the correct number of items of a type, the player will lose a life. When all three lives are lost, the game is over. Completing each level requires taming the creatures on that level. The difficulty of the three levels increases by having harder requirements for 'taming' the creatures, and having the creatures move at a faster speed.

Little 'reaction' images above the heads of the player and the creatures are used to indicate when events take place, like points being gained from collisions with items, and player losing a life or creature being tamed from collisions with the creature (figures 4, 5, 6).

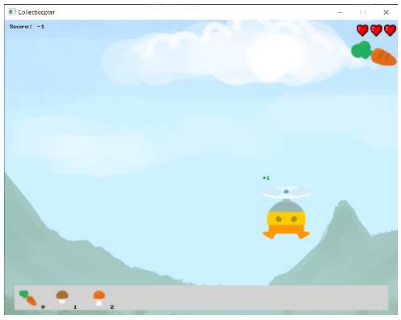


Figure 4. Reaction for points gained

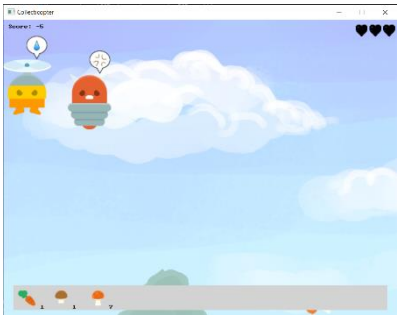


Figure 5. Reaction for creature collisions without required items or score

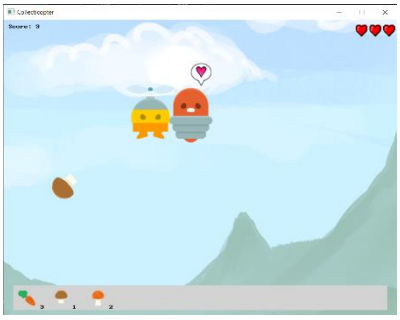


Figure 6. Reaction for 'taming' creature

A start screen is displayed at the beginning of the game (figure 7), and there are two end screens depending on whether the player won or died (figures 8, 9).

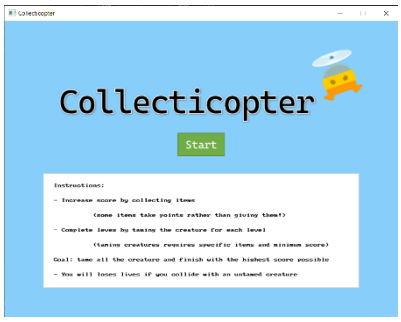


Figure 7. Start screen

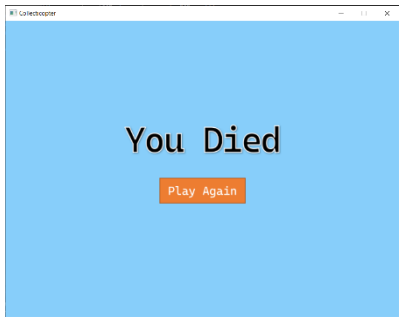


Figure 8. Lose end screen

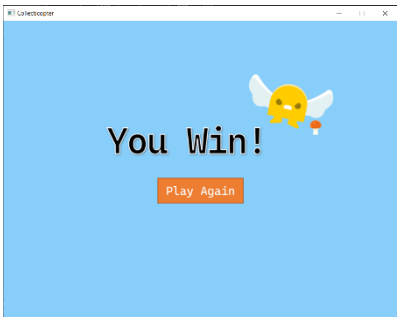


Figure 9. Win end screen

There are screens displayed between the levels of the game indicating that the next level is about to start (figures 10, 11). The conditions that must be met to tame the creatures on each level are also displayed at the top of the screen as a message at the start of the level (figure 12).

Figure 10. Transition screen between levels 1 and 2

Figure 21. Transition screen between levels 2 and 3

Figure 12. Level message

Required Roles

The main classes are *Game*, *Player*, *Item*, *Inventory*, *Score*, *SpriteReactions*, *Reactions*, *Creature*, *SpringCreature*, *SpikeCreature*, *WingCreature*, *Message*, *Title, Button*, *StartButton*, *ResetButton*, *StartState*, *PlayState*, *EndState*, and *TransitionState*.

The game uses the interface *GameState* and four enumerations: *ItemType*, *ReactionType*, *TitleType* and *MessageType*.

Creature is an abstract class that has three child classes: *SpringCreature*, *SpikeCreature*, and *WingCreature*.

Button is also an abstract class that has two child classes: *StartButton* and *ResetButton*.

The Singleton and State Change design patterns are used in the design of this game. The Singleton pattern is used in both the *Player* class and the *Game* class. The State Change pattern is used for controlling the state of the game (i.e. changing between start menu, game play for each level, transitions between the levels, and the end menu).

Table 1: Game <<singleton>> details

Responsibility	Type Details	Notes
Knows current state, backgrounds, player, items on level, level, game over and quit.	GameState, List<Bitmap>, Player, List<Items>, int, bool, bool	
Can handle input, update, draw, check for collisions, remove and bounce items, scroll camera, and reset the game.	RemoveItems & BounceItems take List<Items> Return: void	

Table 2: Player <<singleton>> details

Responsibility	Type Details	Notes
Knows position, velocity, sprite, inventory, score, reactions, life count and hurt.	Point2D, Vector2D, Sprite, Inventory, int, SpriteReactions, int, int, bool	
Can update, draw, handle creature collision, check has required, change image, move, keep player on screen, and reset player.	ItemCollision takes Item, CreatureCollision takes float, HasRequired and move takes string HasRequired returns: bool	

Table 3: Item details

Responsibility	Type Details	Notes
Knows type, position, velocity, sprite, rotation, and value.	ItemType, Point2D, Vector2D, Sprite, float, int	
Can update, draw, check collisions, check on stage, bounce, provide type bitmap and value.	OnStage return: bool TypeBitmap return: Bitmap Value return: int	

Table 4: Inventory details

Responsibility	Type Details	Notes
Knows items, bitmaps, and item counts.	List<Item>, List<Bitmaps>, List<int>	
Can draw, count item types, check has item, and collect and give items.	Collect takes Item and Give takes ItemType ItemCount(typeIndex: int) returns int, HasItem(type: ItemType) returns bool	

Table 5: Score details

Responsibility	Type Details	Notes
Knows points	int	
Can draw and increment points	IncrementPoints takes int	

Table 6: Creature <<abstract>> details

Responsibility	Type Details	Notes
Knows sprite, reactions, timer, required and tamed.	Sprite, SpriteReactiosn, Timer, int[], bool	
Can return sprite and required for type, update, draw, change image, check for and manage collisions, and check onstage.	MySprite returns Sprite, MyRequired returns int[], OnStage returns bool	

Table 7: Reaction details

Responsibility	Type Details	Notes
Knows type, main sprite, and timer.	ReactionType, Sprite, Timer	
Can check if reaction is current, and draw.	Current returns bool	

Table 8: SpriteReactions details

Responsibility	Type Details	Notes
Knows reactions.	List<Reaction>	
Can update, draw, and add and remove reactions from list.	AddReaction takes ReactionType and Sprite, RemoveReactions takes List<Reaction>	

Table 9: GameState <<interface>> details

Responsibility	Type Details	Notes
Can handle input, update, and draw.	Returns: void	

Table 20: Title details

Responsibility	Type Details	Notes
Knows position, type, and bitmap	Point2D, TitleType, Bitmap	
Can return bitmap for type and draw.	TypeBitmap returns Bitmap	

Table 31: Message details

Responsibility	Type Details	Notes
Knows message, type, position, dimensions, and padding	List<string>, MessageType, Point2D, int[], int[]	
Can return position and dimensions for type, and draw.	TypePosition returns Point2D, TypeDimensions returns int[]	

Table 42: Button <<abstract>> details

Responsibility	Type Details	Notes
Knows bitmap, hovered, pressed, and position	Bitmap, bool, bool, Point2D	
Can return bitmap and position for type, update, and draw.	MyBitmap returns Bitmap, MyPosition returns Point2D	

Table 13: ItemType <<enum>> details

Value	Notes
Contains types of all the different items.	*Carrot*, *BrownMushroom*, *RedMushroom*

Table 14: ReactionType <<enum>> details

Value	Notes
Contains types of all the different reactions.	*GainPoint*, *LosePoint*, *Hurt*, *Angry*, *Tamed*

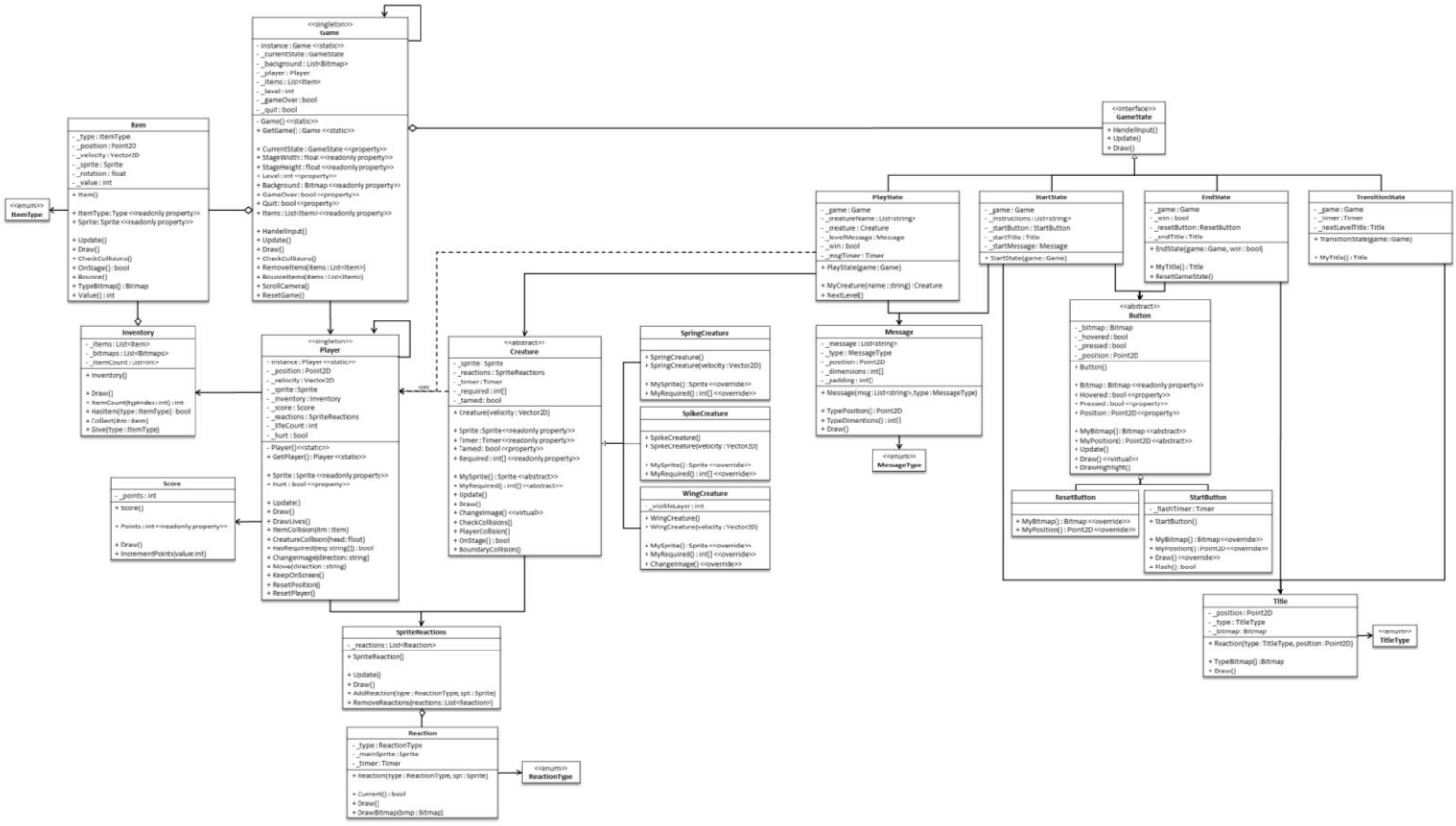
Table 15: TitleType <<enum>> details

Value	Notes
Contains types of all the different tiles.	*Start*, *NextLevel*, *FinalLevel*, *LoseEnd*, *WinEnd*

Table 16: MessageType <<enum>> details

Value	Notes
Contains types of all the different messages.	*MenuMessage*, *LevelMessage*

UML Class Diagram:



UML Sequence Diagram: