26/10/2018

CAB202 Report

Mason Trout

N10170791

Contents

[Feature List: 2](#_Toc528357834)

[Colour Key: 3](#_Toc528357835)

[Functionality Test Plans: 3](#_Toc528357836)

[1.0 Intro 3](#_Toc528357837)

[2.0 Pause Menu 3](#_Toc528357838)

[3.0 Starting Block 4](#_Toc528357839)

[4.0 All Blocks 4](#_Toc528357840)

[5.0 Block spawning pattern 5](#_Toc528357841)

[6.0 Joystick movement 5](#_Toc528357842)

[7.0 Treasure 6](#_Toc528357843)

[8.0 Game Mechanics 7](#_Toc528357844)

[9.0 Advanced Block Movement 8](#_Toc528357845)

[10.0 Advanced Movement 8](#_Toc528357846)

[11.0 Advanced Jumping 10](#_Toc528357847)

[12.0 Food Inventory 11](#_Toc528357848)

[13.0 Zombies 11](#_Toc528357849)

[14.0 Additional Pause Menu Items 11](#_Toc528357850)

[15.0 Potentiometer Block Control 12](#_Toc528357851)

[16.0 Debouncing 12](#_Toc528357852)

[Debouncing Side Note: 12](#_Toc528357853)

[17.0 LEDS 13](#_Toc528357854)

[18.0 Direct LCD Control 13](#_Toc528357855)

[19.0 Multiple Timers and interrupts 13](#_Toc528357856)

[20.0 Program (flash) Memory 14](#_Toc528357857)

[21.0 PWM Backlight Control 14](#_Toc528357858)

[22.0 Pixel Level Collision 14](#_Toc528357859)

[23.0 Serial Communication 14](#_Toc528357860)

[24.0 USB Serial Communication Control 15](#_Toc528357861)

# 

# Feature List:

|  |  |  |
| --- | --- | --- |
| **Assignment Implementation Summary** | | |
| **Item Number** | **Item Description** | **Implementation Level** |
| 1 | Intro 1% | Full |
| 2 | Pause Game 1% | Full |
| 3 | Player size 1% | Full |
| 4 | Block size 1% | Full |
| 5 | Random blocks 2% | Full |
| 6 | Player movement 2% | Full |
| 7 | Treasure 2% | Full |
| 8 | Basic game mechanics 2% | Full |
| 9 | Block movement 1% | Full |
| 10 | Player velocity 2% | Partial |
| 11 | Player jumping 1.5% | Partial |
| 12 | Player inventory 2% | None |
| 13 | Zombies 3% | None |
| 14 | Pause screen advanced 0.5% | None |
| 15 | ADC for block speed 1% | Full |
| 16 | Switch debouncing 1% | Partial |
| 17 | LED warning 2% | None |
| 18 | Direct control of LCD 2% | None |
| 19 | Multiple timers 2% | Partial |
| 20 | Program (flash) memory 2.5% | Full |
| 21 | PWM controlled visual effects 2% | None |
| 22 | Pixel level collision 1.5% | None |
| 23 | Serial communication events 2% | Partial |
| 24 | Serial communication game control 2% | Full |

# Colour Key:

|  |  |  |
| --- | --- | --- |
| Fully Implemented | Partially Implemented | Not Implemented |

# Functionality Test Plans:

## Intro

|  |  |  |  |
| --- | --- | --- | --- |
| **Intro Screen** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| Game starts with intro screen showing name and student number. | Power Teensy and wait for setup(); to run. | Intro screen with name and student number. | Intro screen with name and student number and instructions on how to proceed. |
| When SW2 is pressed the game begins. | Press SW2 on intro screen. | Game starts. | Game starts. |

## Pause Menu

|  |  |  |  |
| --- | --- | --- | --- |
| **When the centre button is pressed:** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| When paused all sprites stop and are unable to move, and is resumed on second press. | Press centre button. | The screen is cleared and game information is displayed. When centre button is pressed again, game resumes. | Screen is cleared and information is displayed and resumes when pressed again. |
| 1. Display lives remaining | Press centre button. | Display remaining lives. | Displays remaining lives. |
| 1. Display current score | Press centre button. | Display current score. | Displays current score. |
| 1. Game time in mm:ss format | Press centre button. | Display time played in mm:ss format. | Displays time played in mm:ss format, only counting when the game is not paused. |

## 3.0 Starting Block

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| Player initialises on a starting block in the top row. | Start game and check where player spawns. | Player spawns on the top row. | Player spawns on the top row. |
| Player sprite is at least 3 by 3 pixels. | Start game and observe player. | Sprite is at least 3 by 3 pixels. | Sprite is 3 by 4 pixels. |

## 4.0 All Blocks

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. At least 2 pixels high. | Start game and observe blocks. | Block sprites are at least 2 pixels tall. | Block sprites are 2 pixels tall. |
| 1. At least 10 pixels wide and easily distinguishable. | Start game and observe blocks. | All blocks are 10 pixels wide and can be told apart. | All blocks are 10 pixels wide and have 10 pixels of space between. |
| 1. Always at least the height of the player + 2 vertically separated. | Start game and observe blocks vertical separation. | At least 2 pixels between the top of the player and the block above. | 5 pixels between top of player and block above. |
| 1. At least 7 safe blocks on screen. | Start game and observe number of safe blocks. | At least 7 safe block on screen. | Between 7 and 18 safe blocks on screen. |
| 1. At least 2 forbidden blocks. | Start game and observe number of forbidden blocks. | At least 2 forbidden blocks on screen. | Between 2 and 13 forbidden blocks on screen. |
| 1. Safe and forbidden blocks are easily distinguishable. | Start game and observe block designs. | Easily distinguishable differences between block types. | Safe blocks are solid and forbidden blocks are clearly different in design. |

## Block spawning pattern

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. Blocks should be initialised in randomly selected rows. | Start game and observe block layout. | Blocks are generated into random rows. | Blocks are generated into random rows. |
| 1. Blocks should be initialised in randomly selected columns. | Start game and observe column layout. | Blocks are generated into random columns. | Blocks are generated into random columns. |
| 1. Blocks don’t overlap other blocks. | Start game and observe blocks. | Blocks do not overlap each other. | Blocks always remain 10 pixels apart. |

## Joystick movement

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. Pressing joystick left and right moves the character one step in that direction. | Press joystick left and right. | Player moves in the corresponding direction. | Player moves in the corresponding direction. |
| 1. If the player is not on a block it will fall. | Move player off block. | Player will fall vertically. | Player falls towards the bottom of the screen. |
| 1. Joystick has no effect on movement when falling. | Attempt movement when falling. | No player movement. | No player movement while falling. |
| 1. If the falling player collides with a safe block, land and stop falling. | Fall of block and land on one beneath. | Player will stop falling. | Player stops falling. |
| 1. No visible pixels can overlap the sprite of a block | Move the blocks into the player while falling. | Player is pushed by block to avoid overlap. | The corner of the player clips through blocks moving at maximum speed. But doesn’t overlap in any other scenario. |

## Treasure

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. Sprite is no larger than player. | Compare sprites. | Treasure is smaller than player sprite. | Treasure is 3 by 3 pixels while player is 3 by 4. |
| 1. Does not overlap any blocks. | Observe treasure movement. | Treasure will not collide or overlap with any blocks. | Treasure does not collide or overlap with any blocks. |
| 1. Spawns in the bottom half of the screen and bounces horizontally from side to side. | Observe treasure movement. | Treasure will move horizontally to one side of the screen and the return in the opposite direction if it reaches the boundaries. | Treasure changes horizontal direction when it reaches the screen boundaries. |
| 1. Stops moving when SW3 is pressed and continues if pressed again. | Press SW3 while game is running. | Treasure will stop moving if it was moving, and when pressed again will continue in the same direction. | Treasure stops on button press and continues in the original direction before being stopped. |
| 1. Disappears on collision with player and rewards two lives and player is sent to starting block. | Collide with the treasure. | Treasure will disappear and give the player 2 more lives as well as respawning the player on the starting block. | Treasure disappears, gives the player 2 extra lives and respawns the player on a random safe block on the top row. |
| 1. Does not reappear unless game is restarted. | Restart the game after collection treasure. | Treasure chest will reappear after the restart. | Treasure only reappears after the restart. |

## Game Mechanics

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. Player starts with 10 lives | Start game and pause to check lives. | 10 lives. | 10 lives. |
| 1. A point is scored when player lands on safe block. | Jump onto a safe block. | Score will increase by 1. | Score increases by 1. |
| 1. If player moves off screen or touches forbidden block, they die | Move player out of boundaries or onto a forbidden block. | Lives will decrease by 1 and player will respawn at a safe block on the top row. | Player respawns on a safe block at the top row and lives decrease by 1. |
| 1. On death, the player respawns on the starting block. | Kill the player and observe where the sprite spawns. | Player will spawn on a safe block on the starting row. | Player spawns on a safe block on the starting row. |
| 1. When the player loses all their lives,   the game over screen allows the player to restart by pressing SW3 and reset all statistics and player position or end the game by pressing SW2 display student number on the screen. | Play until all lives are lost and if the game over screen appears test both SW3 and SW2 and record the result. | Game over screen will display, when SW3 is press on the game over screen, it will restart the game and reset, score, time, and player position reset, and if SW2 is pressed display student number. | Game over screen displays and when SW3 is pressed on the game over screen player statistics reset and the player spawns on the top row, if SW2 is pressed on game over student number is displayed. |

## Advanced Block Movement

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. All blocks move at a constant speed. | Observe block movement. | Blocks will remain equally spaced apart. | Blocks stay at constant speed and distance from each other. |
| 1. Each row of blocks must move in the opposite direction to the row above it. | Observe block movement. | Blocks will alternate direction based on their row. | Each row moves in the opposite direction. |

## 10.0 Advanced Movement

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. Pressing either left or right while on a block allows the player to move fast enough to make progress left or right while in motion. | Try and move left and right on a moving block. | Player can make progress against the direction of the block and moves at a speed relative to the block. | Player makes progress relative to the block on the first and third row of blocks (Blocks that move right). This is due to the DX value of the player being calculated positive instead of negative. |
| 1. If the opposite direction is pressed while on a block, the player stops moving in the direction being travelled. | Move the opposite direction relative the block being stood on. | The player will stop moving while the block is supporting them and fall when it no longer is underneath the player. | Expected result is achieved on the first and third row, but not on second and bottom row, with the player moving opposite to the block direction. |
| 1. If the same direction is pressed while on a block the player progresses at the same speed as the block |  |  |  |
| 1. If the joystick is not pressed when on a block, the player moves with the block. | Stand on a block and observe if the player moves with it. | Player will stay on the block as it travels and not slide left or right. | Only on first and third row of block does the player move with the blocks, if on second or last row, the player slides off. |
|  |  |  |  |
| 1. If falling and land on a block the player will move with the block. | Fall onto a safe block and observe the player sprite. | Player will fall onto the block and move along with it and vertical motion will cease. | If the player falls onto the first or third row of blocks it will move along with it and vertical motion will cease. |
|  |  |  |  |
| 1. All other collisions between the player and safe blocks are consistent with the combined motion. | Check if all collision is consistent with the implemented movement of the player. | Player movement and collision will be consistent throughout the game. | Player movement and collision is consistent throughout the game. |
| 1. Any collision with an unsafe block results in death. | Collide with unsafe block from different angles. | Player will respawn on a safe block and lose a life. | Player respawns on a safe block and loses a life if collided with any side of unsafe block. |

## Advanced Jumping

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. After up is pressed the player will move upwards. And any lateral movement will continue. | Travel in a direction and jump. | Player will travel upwards and arc according to the horizontal movement. | Player jumps upwards but there is no following lateral movement. |
| 1. Once up is pressed the joystick has no effect until the player lands | Jump and try to move. | Player will be unable to move while airborne. | Player is unable to move whilst airborne. |
| 1. Immediately upon landing on a block the player velocity changes to be carried along. | Jump onto a moving block and observe if the player moves. | Player will be carried by the block in motion once they land on it. | Player is carried by the blocks in the first and third row. But is pushed the opposite way on second and last row. |
| 1. If the player jumps off screen the player dies. | Jump off the screen. | Player will lose a life and respawn. | Player loses a life and respawns on a safe block on the top row. |
| 1. Jump velocity should be fast enough to land on the block above. | Jump and land on blocks above player. | Player will move at a speed that allows them to land on blocks above. | Player moves upwards very quickly but is sufficient. |

## Food Inventory

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
|  |  |  |  |

## 13.0 Zombies

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
|  |  |  |  |

## 14.0 Additional Pause Menu Items

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
|  |  |  |  |

## 15.0 Potentiometer Block Control

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| 1. All rows of blocks move at the same speed which is proportional to the potentiometer value. | Slide the right-hand side potentiometer and observe if block speed changes. | Block speed will change relative to potentiometer. | Blocks move at a relative speed depending on the potentiometer. |
| 1. Block movement should range from 0 to a speed that is still clearly visible. | Test minimum and maximum block speed using potentiometer. | Blocks will still be clearly visible and move at an appropriate speed. | Blocks remain clearly visible and move at an appropriate speed. |

## 16.0 Debouncing

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| If switches are effectively debounced. | Play the game and observe if button presses work how intended and don’t cause glitches or flashing. | Smooth button usage and no rapid switching between buttons intended process. | Smooth button use that works as intended. |

### Debouncing Side Note:

The way debouncing was done in this assessment is not what was done in week 9 AMS as it caused the game to run extremely slowly or not start at all with half the screen coloured in. Due to this a method of debouncing using the system clock and a variable that takes the time of the button press and compares it to the timer. If it has been more than one second, the button can be used again but will only run once every second to avoid accidental button presses. The one second delay can also be reduced to any number, but one second was deemed the most practical for general use of the pause menu and other navigation.

## 17.0 LEDS

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
|  |  |  |  |

## 18.0 Direct LCD Control

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
|  |  |  |  |

## 19.0 Multiple Timers and interrupts

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| Multiple timers and interrupts are used to implement the elapsed time. | Either check code directly or acknowledge the play time and debouncing is done simultaneously. | Multiple timers will be used to calculate different useful values such as game time and debouncing. | One timer is used due to performance issues when using multiple interrupts and week 9 AMS, requiring one timer to do the work of two using variables that compare the time of button presses with the play time and total time elapsed since the Teensy was turned on. |

### Timers Side Note:

When multiple interrupts were attempted but running two **ISR(){ };** (Interrupt Service Routines) the Teensy ran significantly slower. Once **if** statements were added to one timer for the use of debouncing and checking the mask against the bit values of the buttons, the game would be so slow it would be longer than 20 seconds between button presses and the game registering them. Both timers were setup correctly. This is why one timer was used for debouncing and timing through the use of variables storing the time the button was pressed adding 1 second and comparing it against the current time, so when one second passes the **if** statement can run.

The game time is the overall time that the timer counts from the beginning subtracted from the difference of the overall time subtracted from the time the pause menu was pressed.

**Difference = total timer – time centre button was pressed.**

**Actual Time Playing Game = total timer – Difference.**

## 20.0 Program (flash) Memory

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| Large objects such as bitmaps are used efficiently and stored so that they take up the least amount of data. | Check code for the used of “static const unsinged char … **PROGMEM**”and the implementation of “load\_rom\_bitmap(… , size of bitmap array);  Followed by: “free(load\_rom\_bitmap(…, size of bitmap array)); | Less memory used on the Teensy. | Less memory used on the Teensy. |

### Programming Memory Side Note:

The method of memory optimisation uses “**static const unsigned char … PROGMEM = {…**” to declare bitmaps so that they are stored in the flash memory. When “**load\_rom\_bitmap(… , bitmap array size);**” is used it loads the bitmap from the flash memory into the RAM. This allows a very efficient usage and storage of bitmaps to minimize the active used memory of the Teensy, making it more efficient.

## 21.0 PWM Backlight Control

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
|  |  |  |  |

## 22.0 Pixel Level Collision

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
|  |  |  |  |

## 23.0 Serial Communication

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| Using Putty with Implicit CR in every LF and Implicit LF in every CR turned on in terminal. Game events and corresponding data are sent to the terminal from the Teensy regarding gameplay. | Start the game, die, collect treasure, respawn, pause or encounter game over to have an event sent to the putty terminal depending on what was done. | A corresponding event and event information is displayed in the Putty terminal based on Teensy gameplay. | A corresponding event and data is sent to the Putty terminal when specified events take place in the game. |

## 24.0 USB Serial Communication Control

|  |  |  |  |
| --- | --- | --- | --- |
| **When game starts** |  |  |  |
| **Test of Specific Functionality** | **Test Setup and Actions** | **Expected Result** | **Actual Result** |
| Pressing “w,a,d” will control player movement, “s” starts the game from the intro screen, “t” stops and starts treasure, “p” pauses the game, “r” restarts the game after game over, and “q” takes the game to the student number after game over. | Play the game through the Putty terminal using the corresponding controls. | The corresponding controls work in controlling the game on the Teensy. | The controls all work the same way the Teensy buttons work, respectfully. |