Diamon runner

Game Design Document

Author: Samuel Savanović

Index

Game Design

Summary

Platformer game similar to Super Mario, collect diamonds but beware more you collect harder it gets.

Gameplay

Gameplay consists of auto-running player character and ability to jump controlled with tap on screen. Player’s goal is to collect as many diamonds as possible while avoiding traps lying on the ground. Gameplay is designed to be hard reminiscent of 90s platformers with perma death and no checkpoints. There are 3 difficulty levels but they only influence point at which player gains speed bonus thus increasing difficulty.

Technical

Screens

1. Title Screen
    a. Difficulty options and start game
2. Game
3. Game over screen

Controls

Since this game is designed for touch devices controlls consists of tap gesture which triggers player jump and autorun. There are only 2 events in game, diamond collected and when specific number of them is collected speed increase.

Mechanics

As all platformer game mechanics consist of simple physics engine that provides collision and object detection and basic controlls.

Performance

Game was tested on desktops using browsers Microsoft Edge, Google Chrome, Mozilla firefox. On androind mobile phone it was tested using Google Chrome, Mozilla firefox and works as expected. There is animation support within game but it causes significant frame drop on some browser so its disabled by default.

Game Flow

1. Player spawns on the hill
2. Starts auto running to the right
3. Close by is first diamond and trap over which player must jump
4. Step 3 is repeated until diamond count is over difficulty setting at which point player will inevitably loose sooner or later at which point he can start again.

Development

Objects

All ingame objects are contained within map object so there is no seperate class for player or any ingame entity, in general everything required to keep trace of game world is contained within it.

1. Map object – contains all information about level including all other objects
    1. Parameters:
        1. Layer – list of layers
            1. data – list of tile index numbers
            2. height – map height, different from screen height
            3. opacity – 0,1, 1 if its opaque
            4. type – is it tile or object layer
            5. visible – boolean, is layer visible
            6. width – map widht different from screen width
            7. x – starting x coordinate
            8. y – starting y coordinate
            9. draworder – specifies drawing order topdown/bottomup
            10. objects – list of objects, each object has following attributs
                1. height – object height
                2. id – objects id, ordinary number
                3. rotation – angle at which object is rotatated in radians

        4. type – type of objects ingame there are player, diamond, trap
        5. visible – whether its visible, boolean
        6. widht – object width
        7. x – x coordinate
        8. y – y coordinate
2. tileheight – height of tiles ingame its 32
3. tilewidth – widht of tiles ingame its 32
4. tilesets – list of tiles(images) with file paths
    1. name – tileset name
    2. tilecount – number of different tiles
    3. tileheight – height of tiles ingame its 32
    4. tilewidth – widht of tiles ingame its 32
    5. tiles – list of tiles in format of index : filepath

Variables

This section contains variables within main.js.

Spritesheet – player sprite sheet
walk – walking animation object
posx – player position on x (used as redundancy)
posy - player position on y (used as redundancy)
diamond_counter – counts number of collected diamonds
difficulty – scalar with which we multiply doom count (0.75, 1, 1.25)
doom – count when double speed is triggered, multiplied by difficulty
moving – boolean, tracks whether player is moving or is collided
backround – backround image
images – array of all tiles specified in map object
running – is game running, false if its game over

offset – offset on y coordinate used to draw world map on bottom of screen: height - map.height * tile height
offsetx – offset on x coordinate for camera, keeps player centered in middle of screen
offsety – offset on y coordinate for camera
jump – is player currently jumping

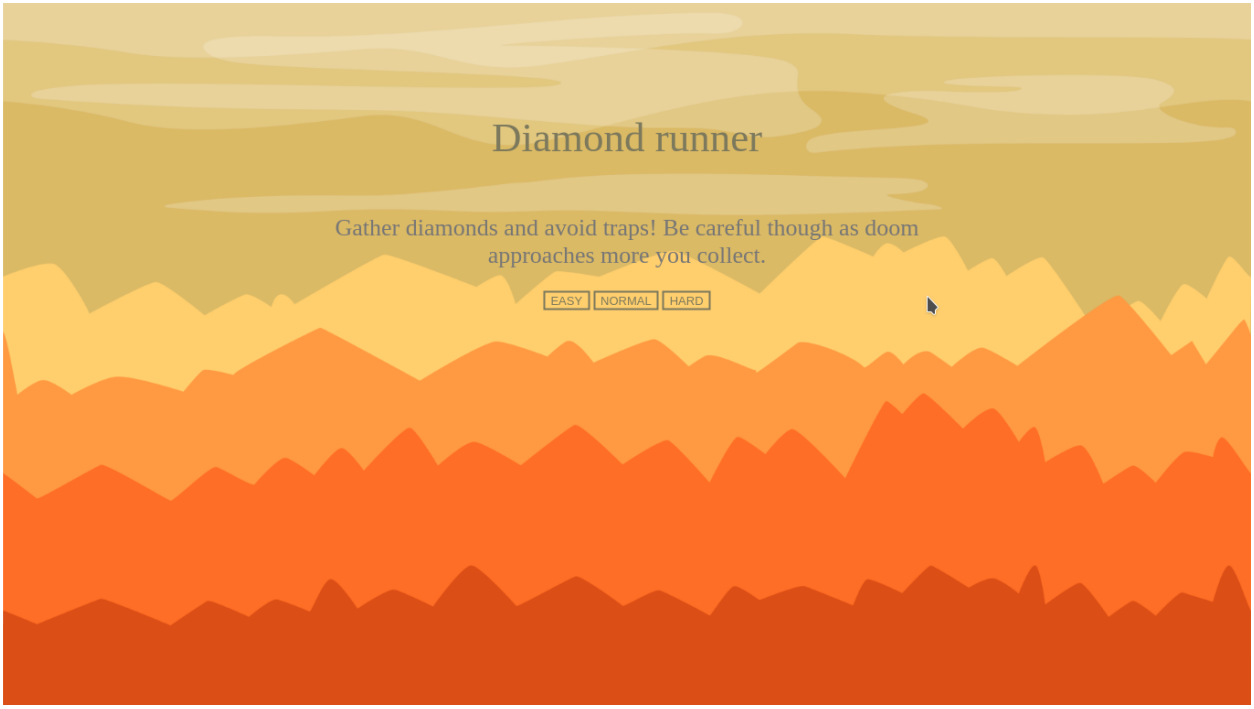
Functions

between(x, min, max) – is number x in range [min, max]
getTile(col, row) – returns tile at coordinate x, y (col, row)
handle_object_collision(x, y) – checks if player collides with object at x,y if he does checks object gid and depending if its trap or diamond does proper action
draw_tile(tile, x,y) – draws tile at x,y
handle_start – triggers jump on tap
collides(x,y) – checks if there is collision with tile at x,y
update() - game loop does the following:

1. calls itself with requestAnimationFrame if game is running else opens game_over screen
2. sets game camera to offsetx + width/3
3. clears screen
4. draws background
5. sets fonts, colors and writes diamond count on screen
6. loops through all tiles in map object and translates map coordinates into screen coordinates and draws tiles at those coordinates
7. loops through all objects in map object and translates map coordinates into screen coordinates and if object is player sets posx, posy but doesnt draw him
8. draws player posx, posy
9. checks and handles player collision at x,y map coordinates
10. implements autorun
11. implements jumping logic

Graphics

Uses tilesets obtained from opengameart all tiles used are freely distributable.




Diamonds: 1
Doom in : 8


Game Over
All good things come to an end. Please try again.
Score: 1
EASY
NORMAL
HARD

Music

Since this platformer is simple it has only one soundtrack in 8bit 90s style.