David Naußed – Platformer Report

Pass:

1) I have implemented on-screen buttons,
   touch controls (bottom-left quarter: move left;
   bottom-right quarter: move right;
   top half: jump;
   and physical controls (d-pad left: move left;
   d-pad right: move right;
   X or A: jump;
2) A static spike enemy which kills the player on collision, plays a sound – Recovery frames are implemented
3) Bouncing coins which use gravity and force to jump. It is a bobbing coin and makes a sound on collection.
4) Simple HUD, it displays the **global** current HP, since the HP is shared among all levels (if player dies, the player gets sent back to level one) and it also displays the coins left in the current level
5) TestLevel is replaced with several levels. Levels can easily get added by putting it in the levels folder. Comments on that:
   a. Due to the scope of this project, tiled needs certain “configurations” for tilemaps to work in the game (LayerType and small changes after it got exported to an XML)
      i. To make your own level, just copy one of the existing levels (preferably level 1) and place walls/ground (preferably just the ones I used since they guarantee correct collider flags) in the static layer and place coins, one player and spikes in the dynamic layer
      ii. After the XML export, open the file and copy the tileset property from another file. It’s nothing special to try it out and I think it is more worth to try out Ruta’s wonderful assets with the Unity Tilemap instead of my creation
6) I added coin and damage sounds, also each level has different music. Music makes use of the MediaPlayer and Sounds make use of a Jukebox
7) The viewport is never going out of bounds, **EXCEPT** if the world is too small, then the viewport clamp prioritises hiding the top and left areas. Prioritising is convenient cause it prevents the camera from snapping back and forth when the player switches sides

8) **Dynamically defined tilesets:** The tilesets get loaded in regards of the path in the tilemap, the tileset has ALL THE DATA, entity type, collider flags, etc. If an entity type from the tileset exists in the data, it gets constructed, also colours are exchangeable

Pass with Distinction:

1) If the player collects all coins in a level, the player gets spawned in the next level. If the player loses all lives, the player gets sent back to the first level. Each level has a different layout, colour and music.
4) The dynamic entities are using their grid location and minimise the amount of collisions with static tiles. Turn the DEBUG_ON flag to true to see the debug interface, which gives a good idea of how the collision works. Green lines are the lines which get checked for collision, everything else doesn't even get regarded with an if-statement. It gets completely ignored
6) **Slopes:** The slopes are pretty, the slopes are working. I spent a long time to make them work and get rid of weird "jittering" and "bumping" when moving downwards. It's a projection of the "next possible" position, with redistributed velocity in regards of the line's direction
7) **Component System:** Transform has positional data, Entity has the components, etc and EntityXML adds the "load XML file" layer. The EntityXML taking care of the data:
   a. WorldManager goes through the dynamic layer
   b. If something in the dynamic layer is not zero, it checks the tile's type and checks whether an entity with this type is supposed to exist, if not it throws an assertion
   c. The XML element with all the data plus the tilemap position get passed to the EntityXML in which the data gets processed and based on the type a config file gets loaded. The XML deals with opening the file and its error handling and afterwards it adds all the components one by one and passes down the needed attributes and data.

Assets:

Art: Thanks to Ruta Sapokaite
Sounds: Thanks to Ruta Sapokaite