Space Blob Advance

Team Information

Team Name: Unidentified

Team Code: 7AB1B234

Team Member 1: Zach Whittle

Team Member 2: Nick Turner

Team Member 3: Henry Robinson

https://www.youtube.com/watch?v=MYxm5ZRwC0I – full game playthrough, please use if you get stuck

Game Overview – Title

Space Blob Advance

Title brainstorm
-space blob
    -space blob uprising
    -space blob descendants

    -the amazing adventures of space blob
-simple
    -descent
    -uprise
    -scale up
    -growth

-terra
    -terra descent

Synonyms
-blob
    -droplet
    -glob
    -splotch
    -ball
    -bubble
    -blot
-creation
    -formation
    -production
    -uprising

-growth

- ***-advance***
- -develop
- -expand
- -increase
- -raise
- -sprout
- -thrive
- -amplify
- -enlarge

-descend

- -crash
- -dive
- -penetrate
- -sink
- -fall
- -gravitate

We wanted our game title to pay homage to 80s era games with a general 8-bit sci-fi aesthetic
We went through many variations of the title which held us back when making a title screen and UI. We wanted our title to be fun and retro while also not sounding goofy. *Space Blob Advance* was derived from a joke where initially, before we had a title, we called it *the amazing adventures of space blob* as it was the craziest and silliest name we could think of, after polishing the name to be less silly but still include the words 'space blob', which we liked.

Game Overview – Game description

This game is about a character who breaks through levels of the Earth to gain access to the Underworld. When the character reaches the Underworld a main 'Boss' is waiting to fight the character.

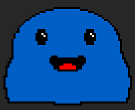
To get through each level of Earth, the character starts as a small blob and gains mass. Once enough mass is gained the character develops the ability to activate a pressure plate to open a door or break through the ground to the next level as it moves towards the Underworld. As the character gains mass and moves through levels it will become harder and more skills in parkour and puzzle solving will be needed. Some of the more detailed levels will have multiple pressure plates or puzzles requiring the blob to split.

Game Overview – Audience

Anyone; our game should be accessible by anyone one with access to and the ability to use a computer. Our game, although challenging will require low physical gaming skill but reasonable problem solving and open-thinking skills. We aim to get our game out to anyone willing to play, no matter their age, gender, or nationality.

Game Overview – Characters and roles

Blobby
The artificially created being whose sole purpose is to bring down a demon who once ruled the realms before going into hiding. Blobby is a blue squishy creature created by scientists who gets lost in space and must make his way to the underworld to defeat the Demon.



Scientists
The 'off screen' scientists that created and tested blobby only to be destroyed in the explosion that starts the player's adventure.

Demon 'numine stilla' Latin – deity blob, power blob.
Centuries ago, a red demon referred to as 'numine stilla' terrorised Earth, committing immeasurable acts of evil and setting Earth back generations. After leaving mass destruction across the globe, the demon retreated into the depths of the underworld. Recorded in myth and legend by different cultures and in different languages through the passage of time one common thread has remained - the pure terror of demon destruction.

Game Overview – Environment

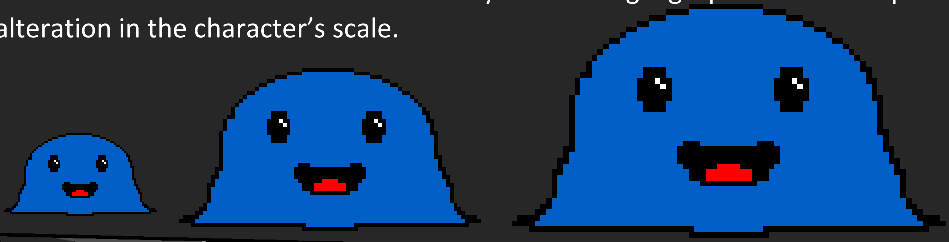
The game will be set in a range of biomes such as

- -space station
- -space
- -forest
- -cave
- -ocean
- -water

Levels such as the space level and the ocean level will have different gravity, this will affect game play and add diversity to how the player must play it.

Game Overview – Theme

The main character will be able to scale up by eating "mass" and scale down by ejecting mass in projectile form and splitting to half its size to active pressure plates and getting into smaller places. The gameplay relates to the theme as the character must be the appropriate scale to progress through the game. We achieved this connection to the theme by introducing logic puzzles that require alteration in the character's scale.



Gameplay and Mechanics – Objectives

The player must return from a failed space mission by smashing through the layers of the atmosphere, earth, caves and into the underworld to defeat the final boss. Short term objectives include gathering mass to increase scale and solving puzzles to break through walls and floors. Long term objectives include budgeting mass throughout the levels and leaving enough mass for the final boss. The player may have to retry levels using the restart function in order to either try different strategies or correctly budget their mass for the rest of the levels.

Gameplay and Mechanics – Perspective

The player will see a third person side on perspective as our game is a platformer. The camera will be smaller than the room and cleanly follow the player without leaving the room using custom camera variables and scripts, different to game makers built in camera functions as they did not suit the characteristics that we liked. To provide a sense of depth we have utilised parallax backgrounds, making use of multiple layers.

The game will be 2 dimensional as we believe it more directly cultivates the genre of games we are attempting to pay tribute to.

Gameplay and Mechanics – Controls

The game will use keyboard and mouse, however if progress on the game is to continue after the video game challenge ends, we hope to implement controller options and touch screen.

A = right
D = left
S = Smash

Space = Jump
RShift = eject
LClick = Split

The game will include several mechanics such as:

Split:
Splitting is used to reduce your mass and to open doors, activating multiple pressure plates simultaneously.

Smash:
The smash option allows the player to smash through the ground moving onto the next stage but will only allow to go through once you have reached a certain mass level.

Eject:
Eject a piece of mass and bring your overall mass down slightly, this ability will be used in the boss fight to enlarge the boss's size until they explode, completing the game.

Gameplay and Mechanics – References

We took heavy inspiration from the 80's platformer era and tried to invoke its visual style but with our own twist such as the asymmetrical geometry of our HUD and UI to help better suit the rest of the game.
We took inspiration from puzzle games such and Fire Boy and Water Girl as it has a fun and relaxing but stressful feel where the puzzles are hard, but you can stop at almost any point, or retry other levels that have already been beaten. The popular online multiplayer game Agar.io was also a loose inspiration regarding the shoot and split mechanic. It has been transformed into a completely different genre of game and has different physical attributes. People may prefer to play this game instead of or alongside those other games if they enjoy a single player experience or prefer games with clearer or more simple objectives than that of Fire Boy and Water Girl. The art style and method of beating the boss non-violently was inspired by slime rancher, where the player feeds the passive bosses until they pop leaving the player with rewards, we wanted to emulate this play style as Slime Rancher is a peaceful indie, strategy simulation game. The terrain art style, consisting of undetailed blocks but detailed decoration was inspired by YesterMorrow, a game we discovered at PAX.

Technical requirements – Platform

The finished game will run on Windows 10, having very minimal system requirements; the game will run on almost any pc or laptop.

Technical requirements – Development Environment

Game maker studio 2
GMS2 will be used to code and export the game as it is a free for education game making platform that allows users to code using the GML coding language. GML is simple and follows logical rules and conventions. Game maker has a wide community allowing us to research and find others dealing with the same issues and bugs that we encounter. It also will enable us to contact GMS2's customer support team.

Fruity Loops Studio (more commonly referred to as FL Studio) will be used to create all of the audio in game. This includes music, possible sound effects or voice lines if needed. FL studio is a perfect tool for sound design and will allow the team to create an immersive audial experience.

Everything seen on screen was created through an online sprite creation tool known as Piskel. It's free to use, easy to learn and allowed for our imagination to seamlessly flow into the game.

Technical requirements – System Requirements

CPU: 64 bit Dual Core CPU
RAM: 2GB
OS: 64 Bit Windows 10
GPU: DX 11
Free Storage: 70MB

Technical requirements – resourcing capability

Hardware:
-laptops
-Keyboard and mouse
-Headphones
-Cameras
Most resources are used for communication, sprite creation, coding, and music development

Software:
GameMaker 2.0, Piskel, FL Studio, Discord, To Do, Teams

Our team will require artists, game designers, testers, coders, and music creators

With the most technical skills Zach will take responsibility for coding, technical requirements, and most information regarding technical processes, system requirements and submission

Equipped with 4 years of secondary art education and years of experience in Piskel, Nick can create incredible sprites that will improve the users' visual experience. Nick's ability to express emotion, theme, and story through 2D animated sprites is an incredible asset for our team.

Having 6 years of practical and theoretical musical experience, Henry will be designated the role of sound design/music. He will also be assisting Nick in the creation of some sprites, had input on the storyline behind the game alongside other general ideas and testing.

Art And Visual – Style

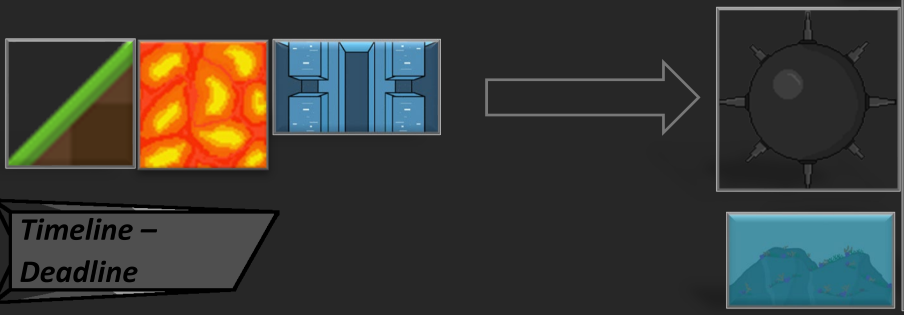
Most of the sprites in our game are formatted in 64x64 resolution with some sprites that needed more detail being 128x128 or 256x256. We have chosen this format as we wanted to try and push ourselves into putting more detail in our work so the outcome will look better. Another inspiration for the 64-bit style was from our game that we created in 2019 "Unidentified", where nearly all the sprites were in 32-bit. Whilst we were reflecting on that game, we realised that we need to increase the quality to create a more aesthetically pleasing game. Using a higher resolution allowed us to convey our thoughts seamlessly into reality and it allowed us to create something that we as a group are proud of.

Backgrounds are in a separate category to most of the sprites as we decided to do these in 1366x768. Backgrounds are very important to the overall impact of the game as they show the player where they are located in the world. As they are high in resolution, they need quite a lot of detail as it is one of the main things people see once entering a game. This proved to be a difficult task at first as we had only drawn in 256x256 at the highest, but once we begun working on it, we discovered the importance of using layers in our designs as they allowed us to separate the background into multiple different layers such as: Background, Midground and Foreground. Using this tool all we had to do was focus on one layer at a time allowing us to make sure that each of the layers were the best we could make them before the submission. This led to us being able to create backgrounds with more ease and detail.

Art and Visual – Process

For the 2021 Video Game Challenge our group has chosen to make a 2D platformer as this was the style that our head programmer was most familiar with and was most sufficient for puzzle games. The art software that is being used for our sprites is called 'Piskel' which is a free online editor for animated sprites and pixel art. As well as being used in our last game, Piskel will be used again this year for all sprites. The style that we are trying to achieve produces clean but detailed sprites as we wanted to add things such as shadows where we think that they would improve the overall look of the game.

Since we have experience in using Piskel, we already knew some of the tools that came with it but once we loaded it back up, we realised that we were only scratching the surface. Throughout creating the game both Henry and Nick advanced their skills on the program and after creating many different sprites they eventually understood all the different aspects and tools of Piskel. This is shown in the sprites as at the start, sprite creation was timely and the result under our standards but once they learnt how to fully utilis the program, sprite making became way easier, as they were able to create detailed sprites in an



Timeline – Deadline

July 22nd
We wish to have our game and game design document finished and ready to submit at the beginning of the submission period so we can submit and not stress afterwards.

Timeline – Timeline

Desired timeline / plan
Register (18/2/2020) → plan 3 ideas → present ideas to target demographic for feedback -> choose one idea (1/3/2020) → work on game and bug test during → first full test (1/5/2021) → implement feedback → Second full test (1/6/2021 → implement feedback → have game finished (1/7/2021)
→ Submit (22/7/2021)

Our initial plan was to have a clear beginning and end of the game development process leaving time in the middle to work on the game and deal with any issues that come up. However, things did not go according to plan

Final timeline
Register (18/2/2020) → 2020 Game challenge cancelled (21/04/2020) → informed that we will not be able to compete next year so terminate progress on game.

2021 Video game challenge begins (18/2/2021)→ informed that we will be able to compete → game brainstorm and selection process → start creating game in gamemaker without built-in physics engine (march/2021) → restart game with inbuilt physics (march/2021) → GAME CORRUPTED (3/5/2021) → game restored (15/5/2021) → Exam period start (25/5/2021) → exam period end (11/6/2021) → return to work less post lockdown → submit game (2/8/2021)

Although it is more unorganised and interrupted than the desired timeline, we managed to produce a game we are still proud of despite all issues we faced.

Timeline – Responsibilities

Although many roles in our team are shared each team member has their own expertise and will mainly work in those areas

Zach

- -Team management
- -Coding
- -overseeing

Zach has a passion for and vast array of skills in coding and game design. As the proposer of The Australian STEM Videogame Challenge at his school, Zach took on the responsibility of overseeing and leading his team as well has helping other Knox teams.

Nick

- -Art

Nick, the Head Artist and story developer in 2019 shared art responsibility's this year as to increase production speed and quality of art. Nick's skills in pixel art help fully translate story ideas and emotion onto the screen. His basic knowledge of GML helped him when he would spend time in video calls with Zach trying to help suggest ways to fix bugs or improve the game.

Henry

- -Music
- -Art

Henry has the ability to create an immersive atmosphere using various instruments and techniques throughout each piece of music. He diverted from the original theme in each piece, keeping the original feel of the song at the centre of the music. By changing the instruments and styles of sounds, he could adapt the music to the style of each level. He also assisted Nick in the making of sprites and backgrounds throughout the game-making process and contributed to the storyline.

To make sure we kept on track we used an application called Microsoft ToDo, where every week we would set dot-points to finish by next week, as everyone was on the shared list anyone could tick a dot point off, and everything would stay organised, and everyone would know what has and has not been done as well as who is expected to do what.

Other Considerations – submission guidelines

We believe before submitting that our game must be the most enjoyable for the user as possible and that we have put in the highest effort. Obviously, before submitting our game we will ensure it follows all rules and guidelines set out by Acer and The Australian STEM Videogame Challenge as well as following out own moral rules such as being of the highest quality we can produce and being a game that we can be proud of and would love to play ourselves.

Our team thoroughly reviewed the Mentor Handbook and Student Handbook to verify that our game follows all rules, we also discuss every major or minor change to the game ensuring we are all happy with the outcome.

Other Considerations – Challenges and Other

This year, our main goal for the Australian Stem Videogame Challenge was to improve on our previous game for the 2019 submission “Unidentified”. Whilst we believe that our standards have risen much higher than they were two years ago, and we have created a better looking, playable and fun game, we faced many challenges that prevented us from putting 100% effort into it.

The first and most significant issue was of course COVID-19 and the surrounding implications that it had on our team and productivity. Having only just recovered from last year's 6-month lockdown, an additional 2 (although much shorter) lockdowns brought the team morale down substantially. There were many points in time where, as a team, we felt as though we were working for nothing as we thought we were entering another long-term lockdown that would delay or completely terminate the competition for this year. Although we still progressed with the videogame, the team and mentors noticed reduced effort. However, we still managed to work without social or face to face interactions via Discord, a video and voice call application. This allowed us to produce artwork and sketch up new ideas by screen-sharing or simply turning our cameras on and brainstorming that way. We had a flexible schedule, however we always aimed to produce 3-5 hours’ worth of collaborative work from home every week, with some weeks exceeding 10 accumulative hours from Nick and Zach. To keep us motivated, we would occasionally set timers of from 20-30 minutes where we can have a break and refresh.

Another complication that significantly impacted our team was a complicated miscommunication of planning last year. When the Australian Stem Videogame Challenge began in 2020, we began our work on the game, however we were soon informed that the competition would carry over to 2021 with the same theme due to covid giving each team another year to work on their game. We were told we would be undertaking other extra-curricular activities during 2021 and therefore would not be able to participate in the challenge. So, adhering this knowledge, we set aside the videogame challenge. Come 2021, the extra-curricular activities that were planned were suddenly cancelled, and so we decided to return to the videogame challenge. We picked up our old ideas but didn't quite like them anymore, so we began once more from scratch, essentially meaning we had lost out on a whole year's worth of progress. This was our largest issue to overcome as it meant we would have to put our heads down and catch up to where other teams may have been at that time. With weekly mentor sessions to monitor group progress, utilisation of Microsoft Teams and To Do, we feel as though we have created a game to our fullest ability with little compromise on our school or social life.

Around 15 weeks into the challenge, another stressful and inconvenient obstacle had risen. Zach began work as normal and opened Gamemaker Studio however no Gamemaker assets, tools, code or created materials would show on screen, nor could he start the game*. No code, sprites or audio were available, and as far as Zach or anyone else in the team could see, there weren't any easy ways of getting around it. GameMaker and Microsoft backups also failed. Zach scoured through many online articles revolving around GameMaker's behaviour with these types of bugs, spent many hours with school I.T, and directly contacted YoYo Games for solutions. Unfortunately for Zach and the team, none of these options turned out successfully. The strangest thing about this one-of-a-kind, undiscovered bug in game maker, behind the user interface of Gamemaker, the raw YML code and file directory looked fine. Being the team member with the most substantial code knowledge, Zach had to carve his own way out of this. The plan was to create an entire new project, recreate every sprite, object, audio asset, room, and script, then copy the text file from the broken project's directory into the new project's directory restarting and reloading the game after every change, the task would be tedious and time consuming but look like the only way forward without recreating the entire game. Zach created an entire new project and sorted through each sprite from the old game, as every room required objects and every object required a sprite, and replicated them in a new project 1 by 1 with almost every sprite replaced the same bug occurred. On the verge of giving up on the game challenge entirely, Zach made a very important discovery; the sprite he had been trying to re-import when the second crash happened was the last sprite to added before the first series of crashes, and so he returned the old game file, removed that sprite from the game entirely, and we were back to normal once more. Nothing stood out about the sprite that crashed the game twice and no one knows why the crash occurred to this day. After almost a month, the team was back in action, and although the Year 10 exams came at an awkward time, we slowly began to make progress.

Other Considerations – Glitches and solutions

Glitches	Description	Solutions
Double y force on jump	When on 2 or more different types of floor and jumping, the character would jump twice as high.	Changing the if statements to else if statements meant the game wouldn't preform both actions at the same time and the player would jump normally.
Trap doors	The trapdoors would only work when the 'right' player was standing on the 'right' pressure plate, meaning you didn't have to just have both activated you would have to have both activated with the right player entity.	Aided by the game maker community we were able to make a workaround that created bullion variables and with statements that allowed the pressure plates to be activated by either the player or the split portion.
Break though	We faced trouble making sure the player had momentum to break through the ground.	By setting short timers to track when the player last jumped, we were able to make it so the player had to jump before smashing through the ground.
Wall climb	When spamming the space key, the player could climb walls or roofs and beat the level incorrectly leading to more glitches down the line when the player would have more mass than they should.	A 30 tick cool down was set after jumping and the player was only able to jump after the timer meaning the player could no longer spam the space bar.
Collision mask	When getting visually bigger the players 'physical collision mask' wouldn't change even if its hitbox and collision mask did, this meant you could clip into walls and exploit other bugs.	Although we were never able to fix this bug, the jump cooldown prevents the player from abusing any glitches it creates.
Animation jitter	The running jumping and standing animation would not move smoothly and would instead randomly change between sprites, this gave the player an inability to walk.	By changing sprite sizes and origin point the player would no-longer flick between spites.
Custom physics	At the very beginning Zach created the game using his own physics, however it	By changing to game makers little known inbuilt physics engine Zach was able to

	seemed the premise of our game would not be achievable using custom physics.	make the game more realistic and smoother than ever. This came with a downside however, as this put the team roughly a week behind after already losing a year. The change also entered Zach into a region of coding he had never explored before, and this meant he had to traverse through a variant of GML that was alien to him.

Other Considerations – Highlights and proudest moments

The moment we got our game back –

On the 15th of the 5th, Zach was away from school sick, one by one he was reimporting sprites, right click, create, “spr_Tree”, save, copy, paste, reload, repeat. Until nothing. The same glitches that started the process, after a small brake down, he realised what sprite had last been imported, the same one that caused the crash initially, heading back to the original directory Zach removed the sprite from the files, and, just like that the game booted up.

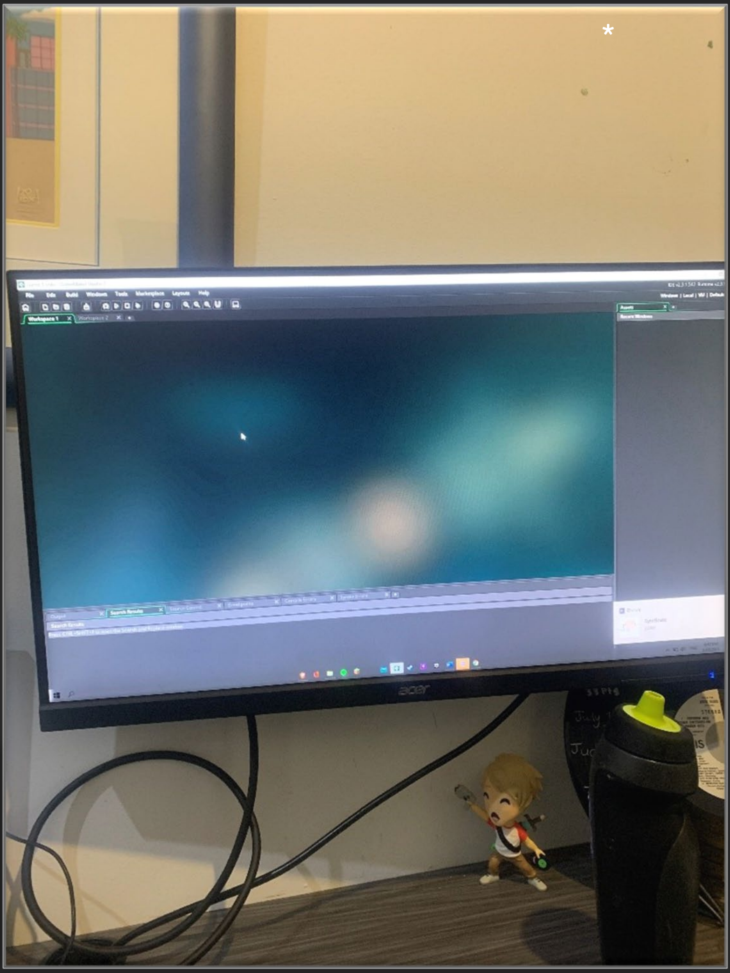
Tile sets –

One of the best and well executed features of our game is the tile sets. This year, using the full function of the auto tile, level creation became as simple as drawing with a brush and then adding solid surfaces where needed, tile sets made level creation easier for the creators and more visually appealing for the player

The bug tests –

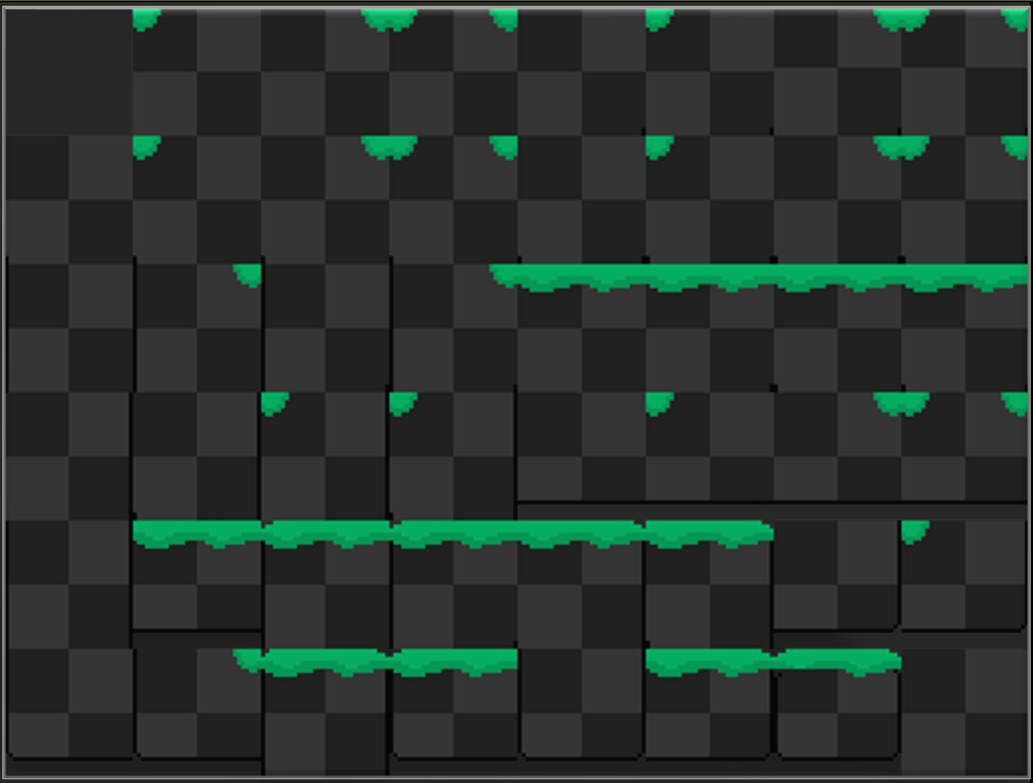
When bug testing with our target audience, they begam playing for hours, competing to get the best time and begging for updates that would allow them to complete the game quicker and refusing to play versions where we patched glitches, they would abuse to skip levels, the first play though had the stuck-on puzzle they thought to be impossible but a couple hours later they would be completing the game in minutes. Seeing people experience such joy from the game we had worked on for month was an incredibly rewarding feeling.

Other Considerations – Gallery

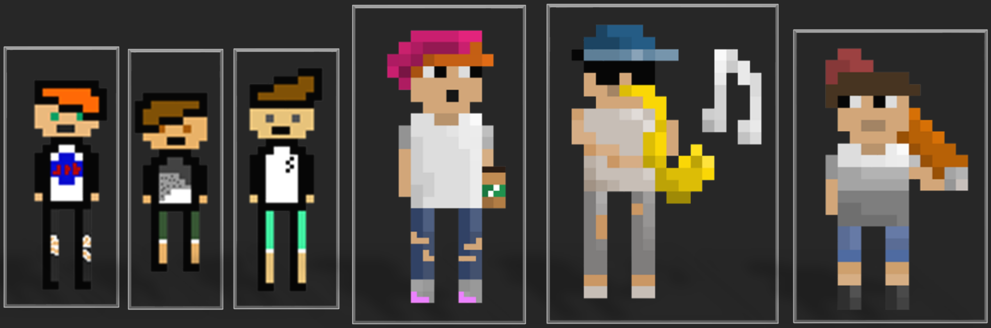


The screen Zach was met with after losing the game - no work or assets found!

Quick concept art and guidelines for the Underworld.



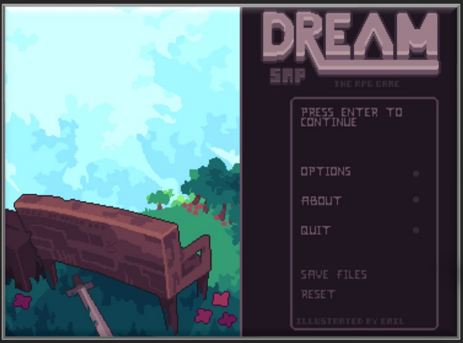
The raw 512 x 384 Tile set sprites before auto tile

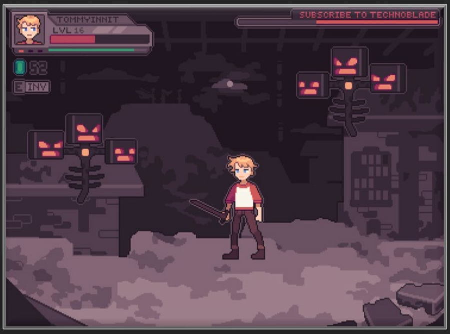


The difference between 2019 and 2021 sprites

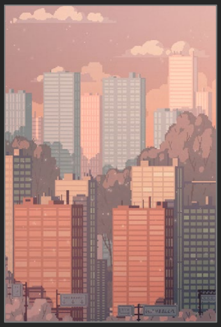
Examples for software we used for communication and organisation.

Some inspiration for the final boss level was MARVEL’s Loki, The Japanese art of kintsugi and Minecraft’s gilded Blackstone.

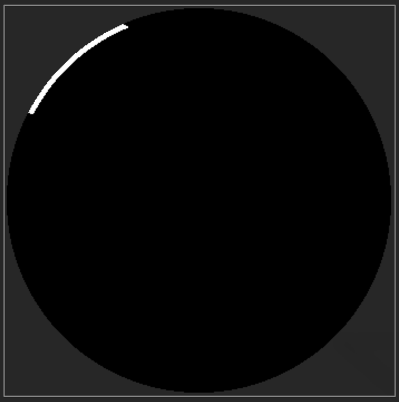


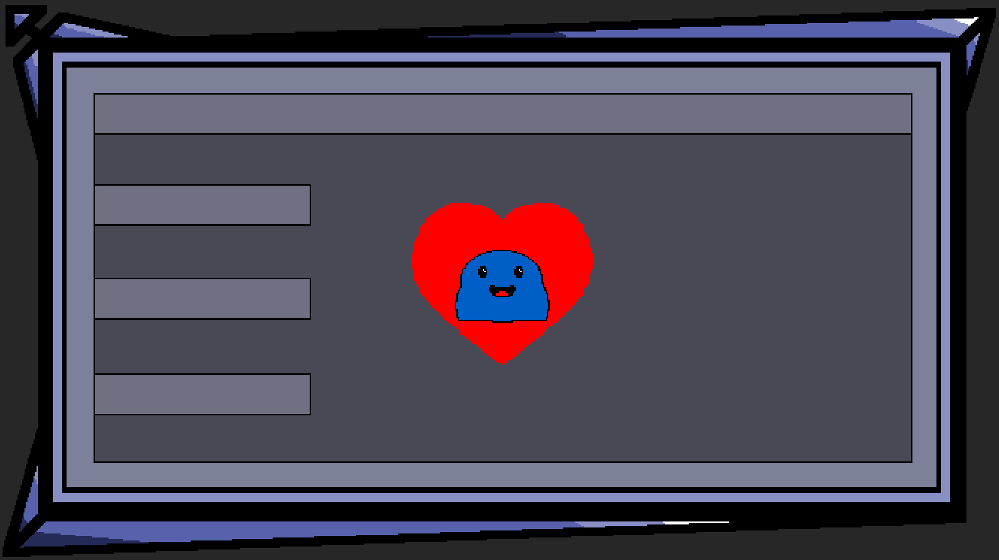


The idea of having much higher quality art for out title screen came from twitter user Emil’s youtuber fanart.



Some more general inspiration from Google images.




Concepts and early renditions of the UI.

https://www.youtube.com/watch?v=-PR0UP5UZys

A cut scene we created for between the tutorial and the space level, however GameMaker does not support cutscenes. The cut scene would have explained the story and seamlessly connected the tutorial to the main story. Consider this a part of the game.

Other Considerations – Code examples

```
Player create

// set up //

// Forces

x_force = 70;


if room != rm_WetCave

{

        y_force = 70;

}

else

{

        y_force = 10;

}



hitground = 0


jumpcooldown = 1


// Controls

go_left = ord("A");

go_right = ord("D");

go_jump = vk_space;



//make sure you cant shoot too much

shoot_reload = 1
```

```
//make sure you cant shoot too much
shoot_reload = 1

// physics //
//rotation
phy_fixed_rotation = true;

// room setup //
//set begining score and keep it throigh rooms
if room = rm_Tutorial
{
            score = 10
}
else
{
            score = score
}

//minimum score for room
if room = rm_Tutorial
{
            passingscore = 11
            rm = "The Begining"
}

if room = rm_Space
{
            passingscore = 14
            rm = "The Abyiss"
```

```
if room = rm_Sky
{
        passingscore = 20
        rm = "Falling"
}

if room = rm_Earth
{
        passingscore = 14
        rm = "Home?"
}

if room = rm_Cave1
{
        passingscore = 14
        rm = "We need to go deeper"
}

if room = rm_Cave2
{
        passingscore = 18
        rm = "Closer..."
}

if room = rm_Cave3
{
        passingscore = 10
        rm = "its getting hotter"
}
```

```
if room = rm_WetCave
{
        passingscore = 10
        rm = "curret affairs"
}

if room = rm_Hell
{
        passingscore = 10
        rm = "glorious purpose"
}

// sound //

if room = rm_Tutorial
{
        audio_stop_all()
        audio_play_sound(snd_Menu,1,12)
}

if room = rm_Space
{
        audio_stop_all()
        audio_play_sound(snd_Space,1,12)
}

if room = rm_Earth
{
        audio_stop_all()
        audio_play_sound(snd_Menu,1,12)
}
```

```
if room = rm_Cave1
{
            audio_stop_all()
            audio_play_sound(snd_Menu,1,12)
}

if room = rm_Cave2
{
            audio_stop_all()
            audio_play_sound(snd_Menu,1,12)
}

if room = rm_Cave3
{
            audio_stop_all()
            audio_play_sound(snd_Menu,1,12)
}

if room = rm_WetCave
{
            audio_stop_all()
            audio_play_sound(snd_Ocean,1,12)
}



if room = rm_Hell
{
            audio_stop_all()
            audio_play_sound(snd_Hell,1,12)
```

```
    if room = rm_Sky
    {
    phy_linear_damping = 1.5
    }
    else
    {
    phy_linear_damping = 0.5
    }
```

Player Step

```
// per tick checks //
//change sprite size with mass
if score < 10 score = 10
image_xscale = score/10
image_yscale = score/10

//go to next room on collision
if (place_meeting(x,y,obj_NextRoom))
{
        room_goto_next()
}

if (place_meeting(x,y,obj_SkyCheck))
{
        if score >= passingscore
        {
                instance_destroy(obj_SkyCheck)
        }
        else
        {
                room_goto(rm_Sky)
```

```
		}
}

// mechanics //

//eject mechanic
if keyboard_check_pressed(vk_shift) && shoot_reload = 1 && score > 10
{
		alarm_set(1,2)
		shoot_reload = 0
		instance_create_layer(x,y,"Layer_Player",obj_BulletMass)
		score -= 1
}

//split mechanic
if score >= 20 && mouse_check_button_pressed(mb_left)
{
		instance_create_layer(x+64, y, "Layer_Player", obj_PlayerSplit);
		score = score/2
}

//brake machanic
if keyboard_check(ord("S")) && place_meeting(x, y, obj_FloorBreak) && score >=
passingscore && hitground = 1
{
		instance_destroy(obj_FloorBreak)
}

// movement //

//move right
```

```
if keyboard_check(go_right)
{
  physics_apply_force(x, y, x_force, 0);
}


// Move left
if keyboard_check(go_left)
{
  physics_apply_force(x, y, -x_force, 0);
}


// Jump Input
if keyboard_check_pressed(go_jump) && place_meeting(x, y + 1, obj_Floor) && jumpcooldown = 1 //&& !place_meeting(x,y-1,obj_FloorBreak) && jumpcooldown = 1
{
//		image_speed = 7
  physics_apply_impulse(x, y, 0, -y_force);
  hitground = 1
  alarm_set(2,10)
  jumpcooldown = 0
  alarm_set(4,30)
}


else if keyboard_check_pressed(go_jump) && place_meeting(x, y + 1, obj_FloorBreak) && jumpcooldown = 1// && !place_meeting(x, y-1, obj_Floor)
{
  physics_apply_impulse(x, y, 0, -y_force);
  hitground = 1
  alarm_set(2,10)
  jumpcooldown = 0
  alarm_set(4,30)

}


else if keyboard_check_pressed(go_jump) && place_meeting(x, y + 1,
obj_PlateWall) && jumpcooldown = 1

{

  physics_apply_impulse(x, y, 0, -y_force);

  hitground = 1

  alarm_set(2,10)

  jumpcooldown = 0

  alarm_set(4,30)

}







//down input

if keyboard_check(ord("S")) && !place_meeting(x,y,obj_Floor)

{


        physics_apply_impulse(x, y, 0, y_force-gravity)


        if hitground = 1

        {

                ScreenShake(0.08*score,60)

        }

}


// animation //

//when in air change sprite and look correct direction

if ((!place_meeting(x,y,obj_Floor)) && (!place_meeting(x, y,obj_FloorBreak))) &&
{


            sprite_index = spr_PlayerAir;

                        if (sign(phy_linear_velocity_y)) > 0

                        {

                                    sprite_index = spr_PlayerSquish

                        }

                        else

                        {

                                    sprite_index = spr_PlayerAir;

                        }

}

//when on grond change sprite to ground

else

{

            if (phy_linear_velocity_x == 0)

            {

                        sprite_index = spr_Player

            }

//when on ground and moving change to moving sprite

            else

            {

                        sprite_index = spr_PlayerRight

            }


}

//change moving sprite to correct direction

if (phy_linear_velocity_x != 0)

{

            image_xscale = (score/10)*(sign(phy_linear_velocity_x))

}
```

```
// trapdoors //


destroy_closed_door= true;


with (obj_Plate)
{
        if (pressed == false) obj_Player.destroy_closed_door = false;
}


if (destroy_closed_door)
{
  with (obj_PlateWall)
  {
    instance_destroy();
         instance_destroy(obj_PlateWall1)
         instance_destroy(obj_PlateWall2)
         instance_destroy(obj_PlateWall3)
         instance_destroy(obj_PlateWall4)
         instance_destroy(obj_PlateWall5)
         instance_destroy(obj_PlateWall6)
         instance_destroy(obj_PlateWall7)
         instance_destroy(obj_PlateWall8)
         instance_destroy(obj_PlateWall9)
  }
}


if place_meeting(x,y,obj_DeathSpike)
{


        room_goto(room)
```

```
          if room = rm_Cave3
                    {
                              score = 18
                    }
          else if room = rm_WetCave
                    {
                              score = 10
                    }
}

if place_meeting(x,y,obj_DeathSpike180)
{
          room_goto(room)

          if room = rm_Cave3
                    {
                              score = 18
                    }
          else if room = rm_WetCave
                    {
                              score = 10
                    }
}

if place_meeting(x,y,obj_Mine)
{
          room_goto(room)

          if room = rm_Cave3
                    {
                              score = 18
```

```
                }
        else if room = rm_WetCave
                {
                        score = 10
                }
}

if place_meeting(x,y,obj_DeathSpikeStone)
{
        room_goto(room)

        if room = rm_Cave3
                {
                        score = 18
                }
        else if room = rm_WetCave
                {
                        score = 10
                }
}

if room = rm_Sky
{
        if score = 20
        {
                instance_destroy(obj_Mass)
        }
}

if place_meeting(x,y,obj_Boss)
{
```

```
                room_goto(room)
}

 Player Draw
 //setup //
 draw_self()
 draw_set_colour(c_black);
 draw_set_font(Fn_Tutorial)



 // tutorial //
 if room = rm_Tutorial
 {
                draw_sprite(spr_Tipbitbigger,1,350,650)

                draw_text(350,630,"Left = A")
                draw_text(350,660,"Right = D")
                draw_text(350,690,"Up = Space")

                                if place_meeting(x,y,obj_TutorialJump)
                                                {
                                                                draw_sprite(spr_Tip,1,640,720)
                                                                draw_text(640,720,"Up = Space")
                                                }

                                if place_meeting(x,y,obj_TutorialJump2)
                                                {
                                                                draw_sprite(spr_Tip,1,900,630)
                                                                draw_text(900,630,"Up = Space")
                                                }
```

```
                    if place_meeting(x,y,obj_TutorialJump3)
                              {
                                        draw_sprite(spr_Tipbiggerbigger,1,1100,507)
                                        draw_text(1100,490,"Eating mass will increase")
                                        draw_text(1100,528,"your score and size, when
you are ")
                                        draw_text(1100,560,"big enough you can split
with LClick")


                              }
                    if place_meeting(x,y,obj_TutorialJump4)
                              {
                                        draw_sprite(spr_Tipbiggerbigger,1,1335,440)
                                        draw_text(1335,420,"when you are big enough
you can")
                                        draw_text(1335,453,"bash through the ground!
Down = S")
                                        draw_text(1335,491,"if you mess up use the
restart button ")
                                        draw_text(1375,529,"in the top left ")
                              }
}


if keyboard_check(vk_f3)
{
          draw_rectangle_colour(bbox_left, bbox_top, bbox_right, bbox_bottom,
c_red, c_red, c_red, c_red, true);
}


if place_meeting(x,y,obj_Tip)
{
          if room = rm_Earth
          {
            {
                        draw_sprite(spr_Tip,1,805,753)
                        draw_text(775,753,"Divide then add")
            }

            if room = rm_Cave1
            {
                        draw_sprite(spr_Tip,1,1522,705)
                        draw_text(1522,705,"Crystals")
            }

            if room = rm_Cave3
            {
                        draw_sprite(spr_Tip,1,202,900)
                        draw_text(202,900,"You're close")
            }

            if room = rm_WetCave
            {
                        draw_sprite(spr_Tip,1,302,614)
                        draw_text(302,614,"Good luck...")
            }

            if room = rm_Hell
            {
                        draw_sprite(spr_Tip,1,1653,2361)
                        draw_text(1653,2361,"Greed is a sin")
            }
}
```