A King’s Lament: The Design Process

Introduction

When I first decided to apply for a Master’s of Science in Game Design, I learned that both Worcester Polytechnic Institute and Rochester Institute of Technology required a portfolio. Now, as a Computer Engineering major with little more art experience than the occasional extracurricular activity or sketch, I was fairly confused by this task. My sister, an art major at the University of Massachusetts in Dartmouth, had submitted a portfolio with all of her recent art pieces, so I assumed that this was what a portfolio was supposed to be. However, I had no art to submit. I could draw, but not well. But upon further research, I learned that not only would I be able to submit more than just art, but I could submit code or an entire game. I even received a suggestion from a graduate student I was asking advice from that I should build an entire game from the ground up. So, with three weeks until the application deadline, I set out to do so.

Initial Design

Deciding the Game Type

First, I needed to decide what game I would make. I decided early on in the planning process that I wanted to make a game that people would enjoy playing rather than one that would just showcase my ability as a game designer, so I had this in mind while I was coming up with the design. I had an idea bouncing around in my head for a while of a game where the player fights off endless waves of monsters in their mind that were meant to represent the character’s anxiety. I thought about this idea for a while, and was very close to making it, but I decided finally that while it would be an excellent game to show my passion for allowing players to feel empathy towards others, I didn’t feel I could do it justice just yet. I had really wanted it to be a Dynasty Warriors-style game, and as one person with about three weeks to make a game, I did not think that was feasible.

So I turned my attention to some games I had made my freshman year of college. These were rudimentary ones, meant to help me learn how to code in C and C++, and consisted of a paint program and a Space Invaders-style game. These would be easy, minimal graphics and not much coding to do, but I wanted to really build something from the ground up and tell a good story through this game, so I decided against that eventually.

Finally, one of my friends suggested I try a simple platformer. This would give me the ability to tell a story, but would not be so difficult to code that it would take me longer than two weeks. In fact, platformers had been made at many 48-hour game jams, so this would be a perfect game to showcase my ability while still finishing it within a deadline. I also decided I would make this game using Unity since I had never used it before and wanted to learn how to.

Deciding the Game Style

Now that I had the type of game I was going to make, I needed to figure out the details. What would be the premise of the game? How would the player win? Where would narrative come in? How much narrative did I want in this game? What kind of graphics did I want to use?

I decided early on in the planning process that I wanted this game to be a retro, 8- or 16-bit style because a few of my friends were on a systems engineering team building a retro video game cabinet that would advertise games built by students in the computer science and gaming and animation departments. One of them suggested I build a game for the cabinet, and I decided that this was what I would do.

Next, I did a little brainstorming about the premise of the game. Why would a person be jumping platforms and moving around a space? Then I remembered that I had explored lots of ruins when I lived in Turkey for two years. My dad had been fascinated by them, especially by Hittite castles, and so he would drag us all around the country to look at different ruins. My sister and I were less than impressed at the time, as Turkey was really hot all the time and we didn’t really like walking around outside, but now I am so grateful for these experiences. One of the castles my dad took us to was called Kizkalesi, also known as the Castle on the Sea or the Maiden’s Castle. This one was actually built around the time of the Byzantine Empire, and it has a legend connected to it that I have always been fascinated by.

According to the legend, a king in the land had been told by a fortune teller that his daughter would be killed by a snake. So, he built a castle on the sea because there were no snakes there, and he sent his daughter to live there so he could protect her. However, a snake had hidden in one of the fruit baskets that was being brought to the castle, and the princess died from its bite.

This story, sad as it was, has beauty to it and has stuck with me for the better part of 10 years, and so I decided to use this story as the basis for my game. From there, I needed to decide how to incorporate this story into the game. I decided that it would be interesting to play a person exploring the ruined castle, and then learning the story either from NPCs or plaques along the way. But this was little more than an interactive history book, and while those have their merits, I really wanted to do something more with the game to make it fun for a wide range of audiences.

I toyed with the idea of having snakes around the castle for the player to fight, but this didn’t seem like it had much of a point. After all, if the player is exploring ruins, then the princess is already dead, so there is nobody to save, other than themselves, at which point they should just leave the castle. It didn’t really seem like there was really any need for fighting at all, so I quickly decided against allowing the player to attack anything at all.

Something struck me about the story though: how sad the ending was. It made me consider how the townspeople, and more than anything, the king, would feel when their beloved princess was dead despite all their efforts. I wondered if these people would even haunt the ruins, forever lamenting their fallen royal. It would especially make sense for the king. From there, I thought about how I could give this sad story a happy ending. Restore the king to his daughter and allow them to leave for the afterlife, satisfied. But I still wanted this exploration aspect of the game.

Thus, I decided that the player would start by arriving at the ruins of the castle, finding plaques with the story of the king and the princess throughout the level. Eventually, they would arrive at the lowest part of the castle, where they would find an artifact that would summon all the ghosts to the castle. From there, the player would need to navigate the castle again, now with ghostly entities blocking their way in places, to the top of the castle, where they would find a second artifact that would finally appease the spirits and reunite the king with his daughter. There would be a brief cutscene, the king would thank the player for reuniting him with his daughter, and the father and daughter would leave together.

At some point, I also decided on a name for the game, which seemed fitting for the story: A King’s Lament.

Art

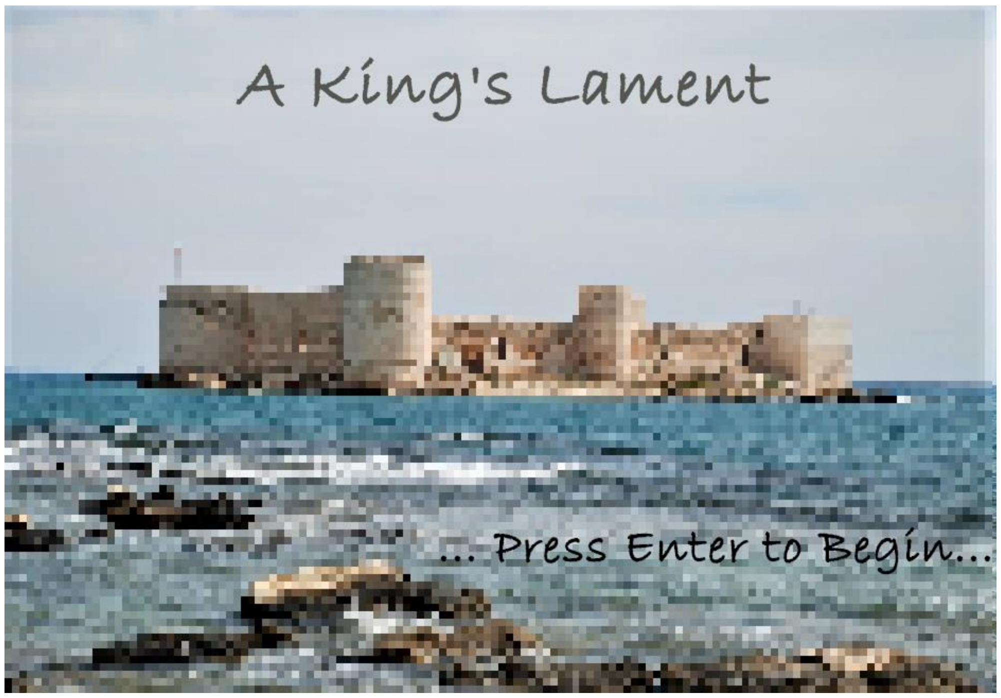
As I mentioned before, I am not an artist, so I was fairly hesitant about this part of the game. I had decided on 8-bit characters to minimize the amount of artistic skill I needed, but there were still lots of sprites, items, and backgrounds that needed to be made, pixel by pixel.

I decided to use GIMP as my art program because it worked pretty well on a pixel level. I didn’t want to use something like paint because it blurred pixels after saving the image. I would have liked to use a program like Photoshop, but I did not have a license for it. So I thought GIMP would work just fine, and for the most part it did. Around the end it began blurring pixels when I tried to scale things up from the small, pixel art level, but at that point I had enough art that was the size I wanted it to be to just stop scaling pieces.

What I did to draw each piece was to create a small version pixel by pixel and then scale it up. This way, it had that 8-bit look but wasn’t too small for Unity to handle.

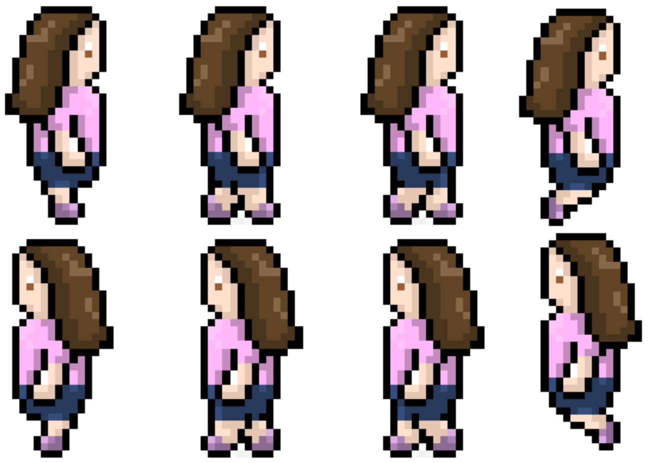
When I started designing the characters and backgrounds, I wanted to make sure I was keeping the actual Turkish castle in mind. So I looked up the castle, and continued to do research throughout the process. My father loves Hittite castles, so I assumed Kizkalesi was a Hittite castle, but I was surprised to find it was built by the Byzantine empire. With this in mind, I did my best to keep Byzantine architecture in mind while designing the castle.

I also decided to make the main screen a pixelated picture of Kizkalesi with the game title and a simple start menu. For this, I found a picture of Kizkalesi on Wikimedia commons that allowed editing and I ran it through a pixilation website. I made sure to follow the CC license in the end credits of the game.



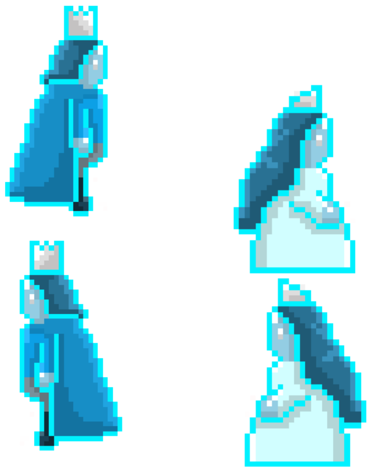
Sprites

I did not want to spend too much time on character design, so I decided to use myself as the player character. I thought it was a bit of a cool touch since I had explored this castle before. The player character was the first sprite I made, and I gave her pale skin, long brown hair, golden brown eyes, a pink t-shirt, jean shorts, and purple shoes. I used online tutorials to learn how to do pixel art shading, and I'm fairly impressed by the results. I also gave the player character a few simple animations to give some life to the game.

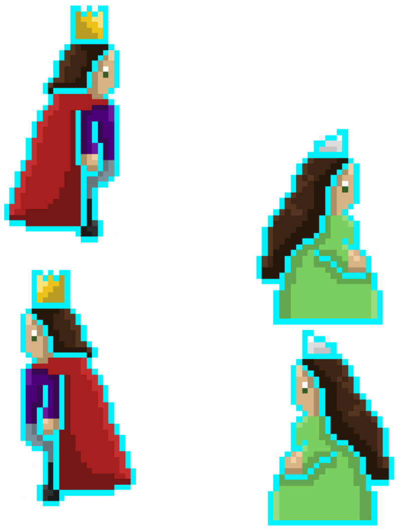


Speaking of adding life to the game, next I needed to design the ghosts. I knew I wanted to give the king and princess their own sprites, but I wasn't entirely excited about designing multiple townsperson sprites. Instead, I decided to make the townspeople sprites little puffs of smoke. If I had more time and was better at animating, I would have made these swirl like smoke as well, but I was eager to continue with the design, so I just made one stationary sprite.

Next, it was time to create the king and the princess. For the king, I decided to put him in a shirt, pants, boots, and cape. I gave him shoulder length hair and a crown on his head. For the princess, I placed her arms in front of her instead of to the sides like the king and the player character, and I put her in a long dress. I also gave her very long hair and a tiara. Coloring the king and princess were interesting because during the planning process I had thought it would be interesting to have a mechanic where the ghosts would be in mourning until the player finds the princess' crown. With this in mind, I decided to have two models for each character: one entirely in blue for their mourning ghosts, and one in full color with a blue outline for their appeased ghosts. The mourning ghosts were fairly easy to make.

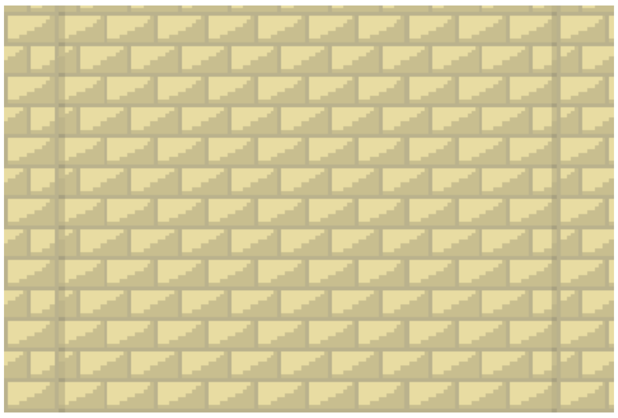
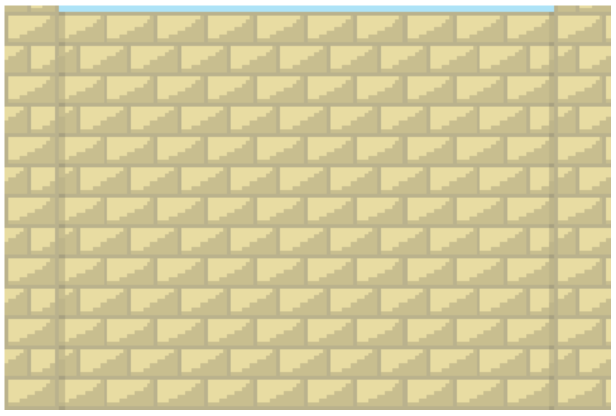


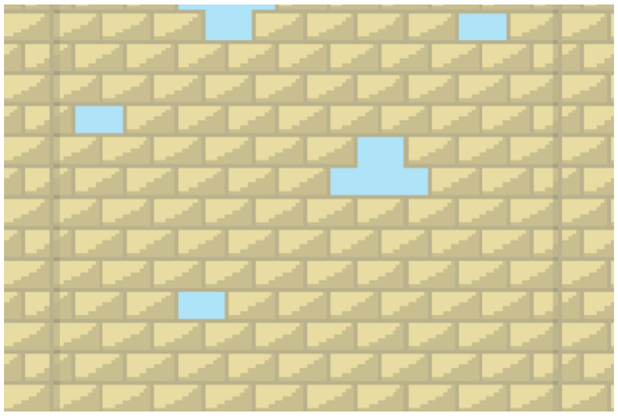
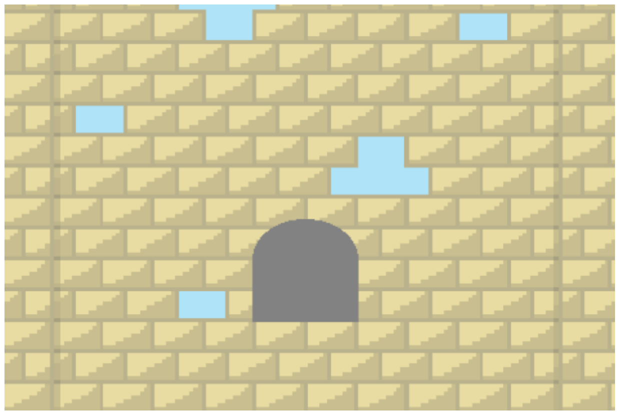
The appeased ghosts were a little harder. The king was easy because I went with fairly normal "king's colors" for his outfit: I made his shirt violet, his cloak red, his crown gold, his pants grey, and his boots black. For the princess, I had a hard time deciding whether to make her dress violet as well or give it a unique color. I didn't really want to use blue because there was already so much blue in the game to symbolize that they were ghosts, so I wanted something that showed that she was free to go to the afterlife. I also didn't want to use pink because I didn't want her to match the player character's shirt. So eventually I went with green, and kept her tiara silver. For their skin tones, I decided to make both the king and the princess look like the Turkish people who live around that area: brown skin, dark brown hair, and green eyes.



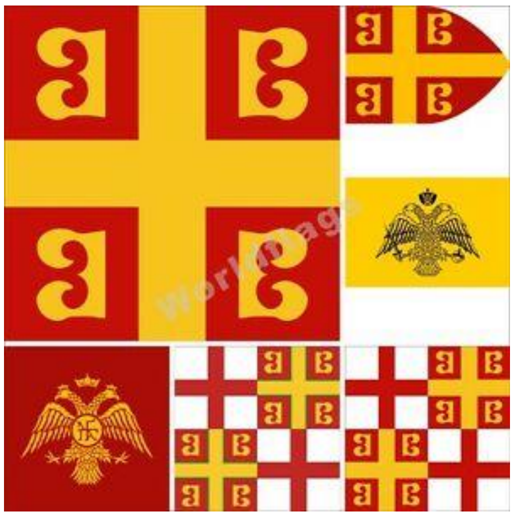
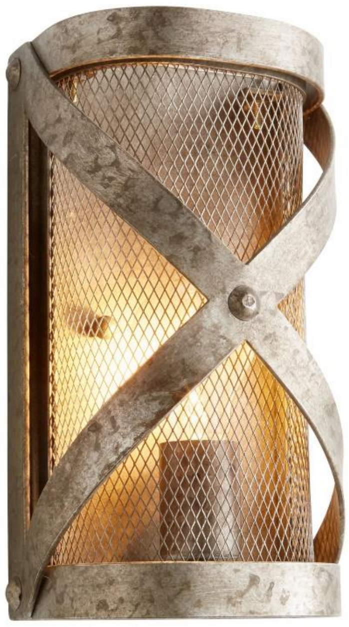
Backgrounds

Kizkalesi is a stone structure that has a light tan color to it, so I decided to make brick walls of that color. I made pillars on each side of each room, and deleted the top row of bricks and colored it light blue to denote an outside courtyard type area. I also wanted to make the walls look somewhat crumbled, so in random areas I deleted bricks and colored the empty space sky blue. I did about four iterations of the inside walls and two of the outside walls, then put a door in one of them to go between upstairs and downstairs.

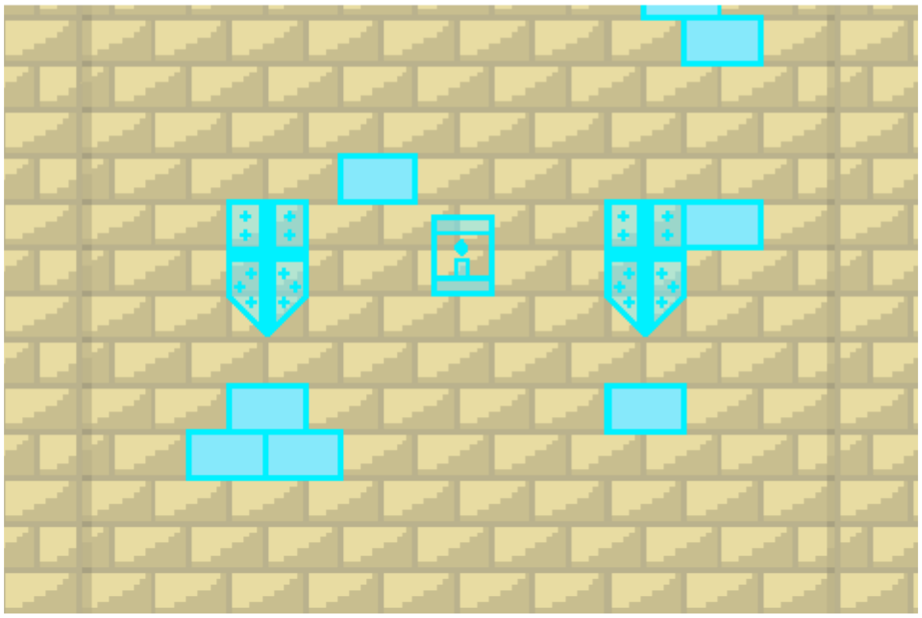



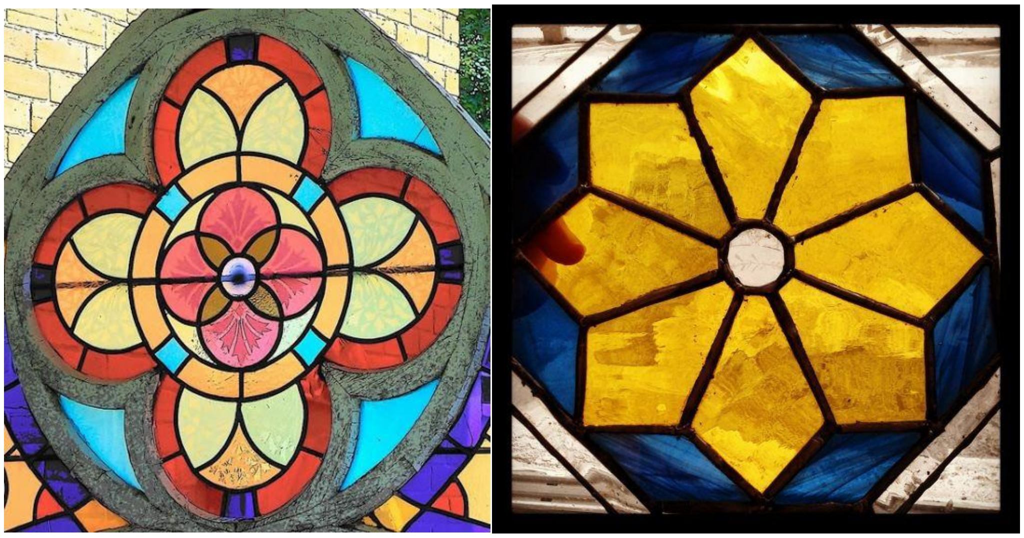
For the ghost backgrounds, I wanted to make them look like they would have when the people of the castle would have lived there. So like the king and the princess, I decided to outline the bricks in the same blue color to show where bricks would have been. Then I decided to make flags and sconces to decorate the walls. Before doing so, I looked up Byzantine flags and sconces, and found these:

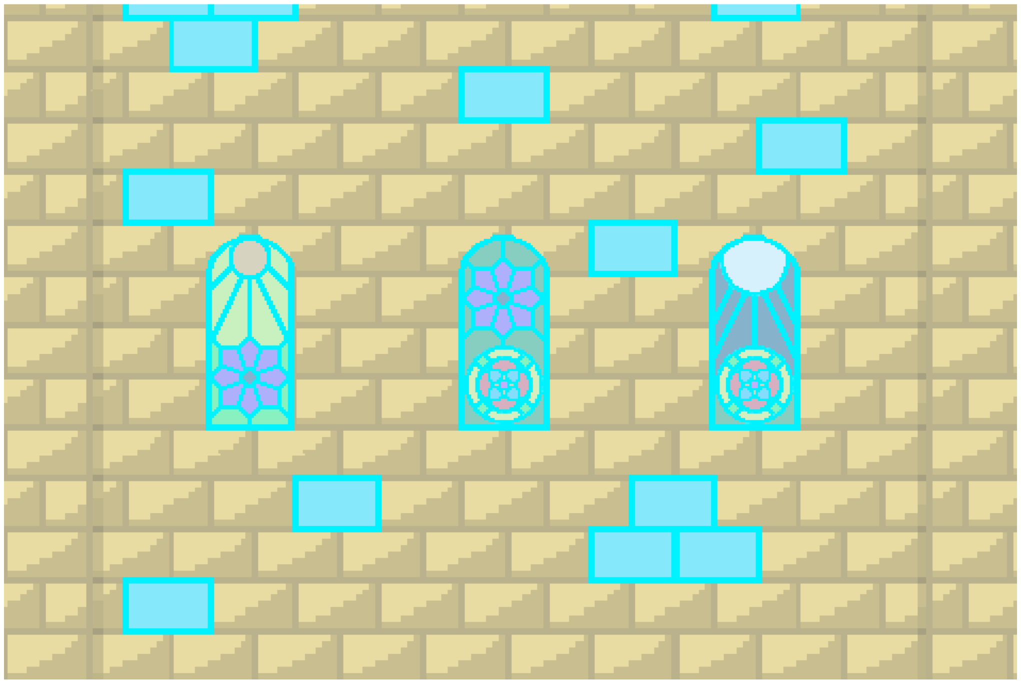
I did not feel like I possessed the ability to do either of these designs in pixel art, so I modified them to be a bit simpler and then placed them around the middle of the walls.



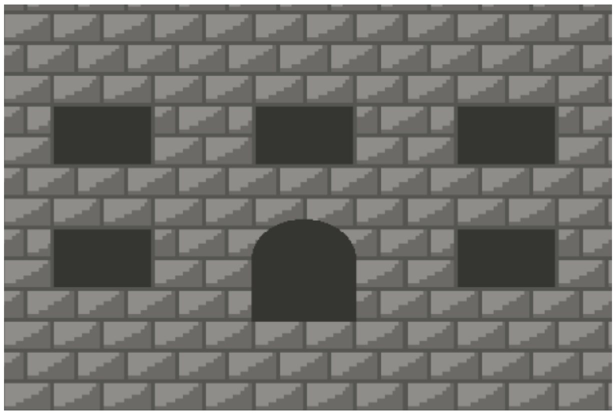
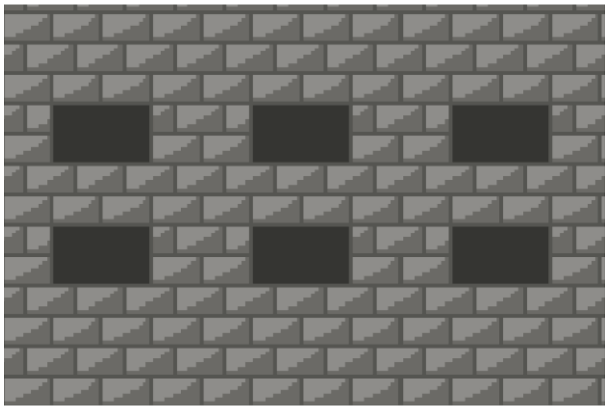
For the princess' and king's room and the throne room, I decided I wanted to do a stained glass window. So I made a single, small arched window in the middle of the wall for the king's and princess' rooms, and then I placed three in the throne room to add regality to it. Next, I did a little bit of research into Byzantine stained glass windows. I found these:



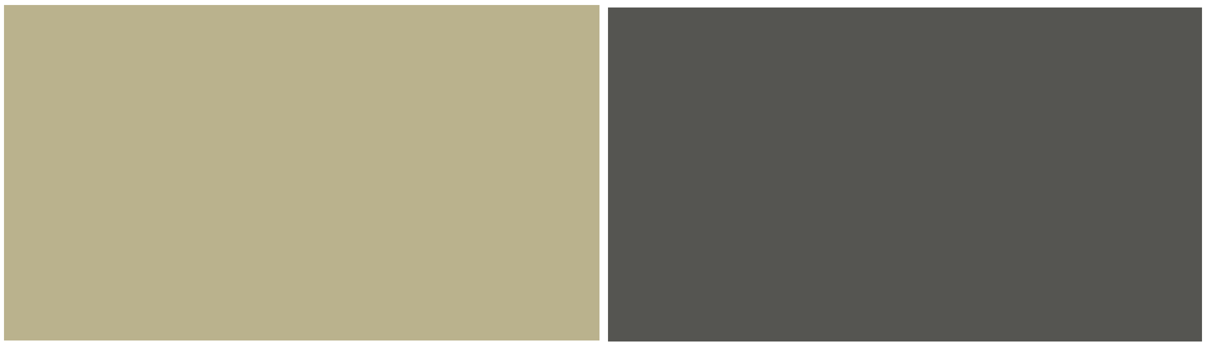
I decided to do a simplified version of the left flower to place on the king's window, and then place the right flower on the princess' window. So I put these flowers at the bottom of the windows, and then put a sun shining down on the princess' flower and a moon shining down on the king’s flower. Finally, I put the king’s flower on top and the princess’ flower on the bottom for the third, middle window of the throne room.



Finally, I decided to do a fairly simple design for the crypt. I took the same brick wall that I had been using for the rest of the level and colored it grey, then added large boxes where coffins would go. I removed the bottom center box and added a door for the start of the crypt.

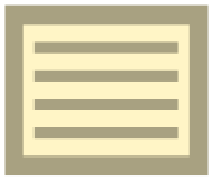



For the floors, I decided to go extremely simple and take the line color for the normal walls and the crypt and make a solid color of that. That way, I could resize it without worrying about warping.

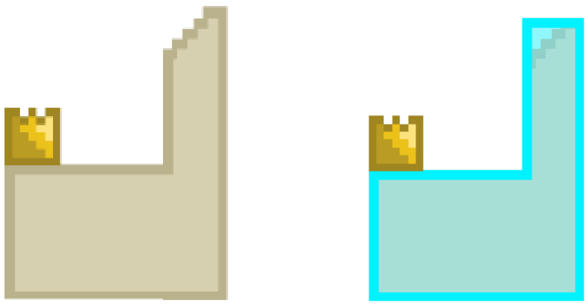


Items

I wanted to make the items fairly simple, because at this point I had been working on art for longer than I was comfortable with it. For the lore tablets, I made a rectangle with a 3:2 aspect ratio and put a few lines on it to denote writing.



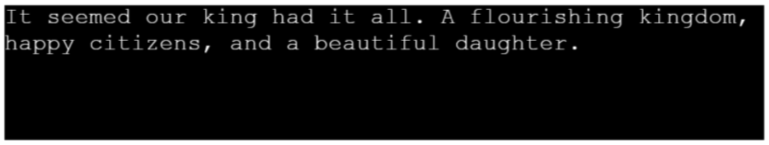
Next, I decided to make two versions of the king's throne, one when the player finds it and one when the player interacts with the crown and summons the ghosts. For this, I decided to make the throne a similar color to the walls and slightly broken on top, and then made a second version highlighted in blue. I then copied the crown from the king's sprite and placed it on the throne.



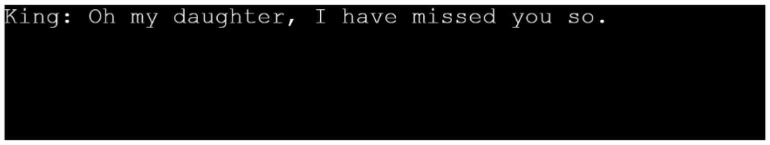
For the princess' casket, I followed a similar pattern and made the stone the same color as the rest of the crypt, then copied the tiara from her sprite and placed it on the left side of the coffin.

Dialogue

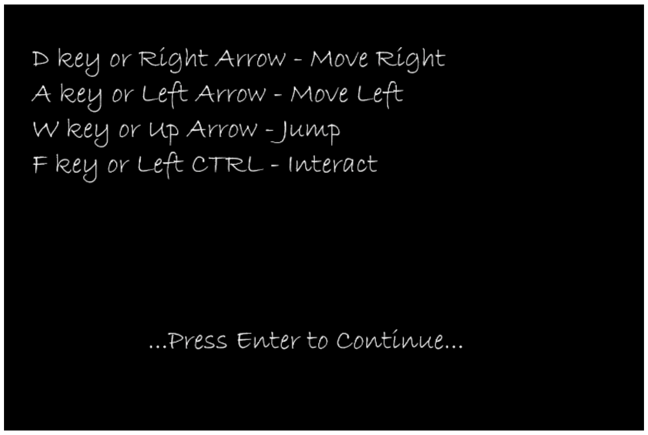
I decided to make the dialogue boxes extremely simple by making them black rectangles that fit at the bottom of the screen with white retro-looking text. For the menus, I decided to use Bradley Hand ITC to add a medieval feel to the game.



```
It seemed our king had it all. A flourishing kingdom,
happy citizens, and a beautiful daughter.
```



```
King: Oh my daughter, I have missed you so.
```


D key or Right Arrow - Move Right
A key or Left Arrow - Move Left
W key or Up Arrow - Jump
F key or Left CTRL - Interact

...Press Enter to Continue...

Music

I used *The Legend of Zelda* as inspiration for the music of this game. I like the regal feel of the main theme, and "Midna's Lament" from *The Legend of Zelda: Twilight Princess* has a sadness that I wanted to capture in the music of this game too. I actually had "Midna's Lament" in mind when I came up with the title of the game. But I did want to have a completely original soundtrack for this game. I also wanted to use chiptunes or something similar to add to the retro feel. Also, while I know the player will only hear the music, I decided to name each song something relating to the situation in case I ever wanted to make this into a soundtrack.

I pulled out GarageBand on my iPad and started playing around, and eventually I came up with a melody that had the same feel as "Midna's Lament" but seemed unique enough to me. I also sang it into a song recognition software to make sure I didn't subconsciously write a song I already knew, and it didn't recognize anything, so I kept going. The song started on a C and used 3 flat notes, so I decided to make it in the key of C minor, and I decided to make it 120 BPM because I had some chord progression in mind to that tempo. I gave it a ¾ time because it seemed to follow that pretty well. From there, I played with the chord generation software on my iPad to create 8 distinct melodies, all building off of each other. The chord progression for all of the songs is C minor, G minor, F minor, G major, C minor, G minor, F minor, B flat major, C minor.

The title song has the main melody plus a base line and a downward arpeggio from the fifth of the scale to the root of the scale. The arpeggios are each one beat long. This is supposed to be a song reminiscent of “Midna’s Lament,” a song showing the sadness of the king over losing his daughter. I named this song “A King’s Lament,” since it was the main theme of the game.

The song that plays while exploring the castle is one beat of the base line one sixteenth note arpeggio per chord going from one fifth to the fifth an octave higher. This track makes use of the silence, and the notes echo in a way that fits the haunted status of the castle. I called this song “Exploration.”

While exploring the castle with all of the ghosts, I decided to add the downward arpeggios from “A King’s Lament” to begin building to the conclusion. For these, I shortened each note to an eighth note and added a rest for half a measure to continue this use of eerie silence. I titled this one “The Past Returns.”

For the next four songs, I had specific additions to each exploration song for the king’s and princess’ rooms to set them apart. For the king’s room, I put the melody back in, but with each note turned into an eighth note “bop” note in place of the original melody. For this, I meant to add a premonition of the king’s lament while still making use of the silence that permeated the rest of the levels. I named the track for exploration without ghosts “Regality” and the one with ghosts “Alone,” since the player will find the king’s ghost facing away from them, alone.

Finally, I decided for the princess’ room, I would add a downward sixteenth note arpeggio leading into the upward arpeggio of the next chord. I named the track without ghosts “Emptiness” to denote the sadness of the princess’ empty room, and the one with ghosts “Hope” to show the princess’ hope for freedom.

Finally, I decided I wanted a song for the final cutscene of the game. This one had mostly the same parts as the original theme, but I shortened the downward arpeggios to eighth notes and pushed them together to make the song seem faster, and then added higher-pitched sixteenth note arpeggios going from the root of the scale to the next root an octave up and back down. These notes continue moving up and down through the entire piece to give this excitement that I wanted to convey through the piece, but I kept the whole song in a minor key to maintain continuity. I titled this song “A King’s Reunion,” and while it is an extremely busy song, it is my favorite of the eight songs I made.

I had two more places, the throne room and the crypt, and I decided that these should reflect the gravity of the objects, so I didn’t compose any music for these locations in favor of absolute silence.

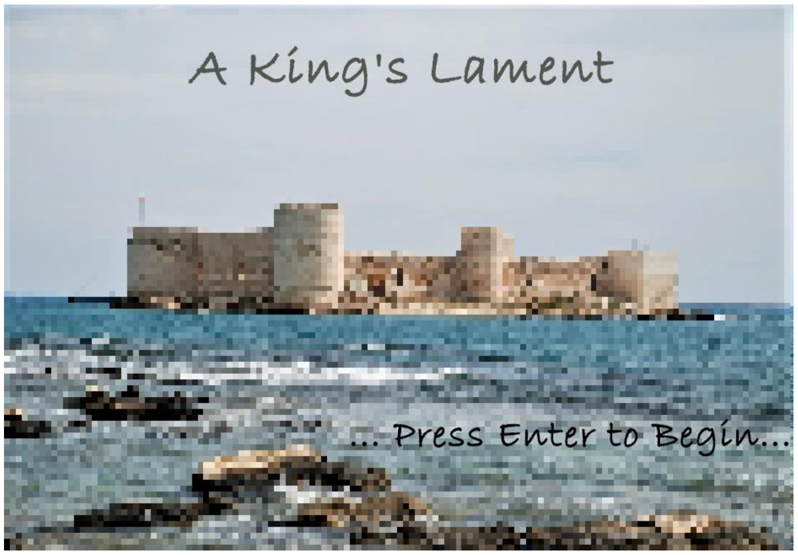
Level Design

I decided to have two basic levels: exploration without ghosts and exploration with ghosts. However, I decided to do a few scenes for each level because I wanted to change music but didn't want to add triggers to change it. Later I ended up using triggers for other pieces of the levels, so I probably should have just had one scene for each level. Whatever the case, I decided the levels would consist of 6 scenes each, and then there would be a title screen, controls menu, a final cutscene, and an end credits screen.

For the design of the castle itself, I decided to have some basics of classic video game castles. While Kizkalesi itself is completely outdoors, I thought I would make a space that would work narratively for the game. So, I decided to have three floors: a downstairs, an upstairs, and a below-ground crypt. There would be three special rooms: a throne room and the king's and princess' rooms, and then an outdoor area leading into the castle to stick to the source material.

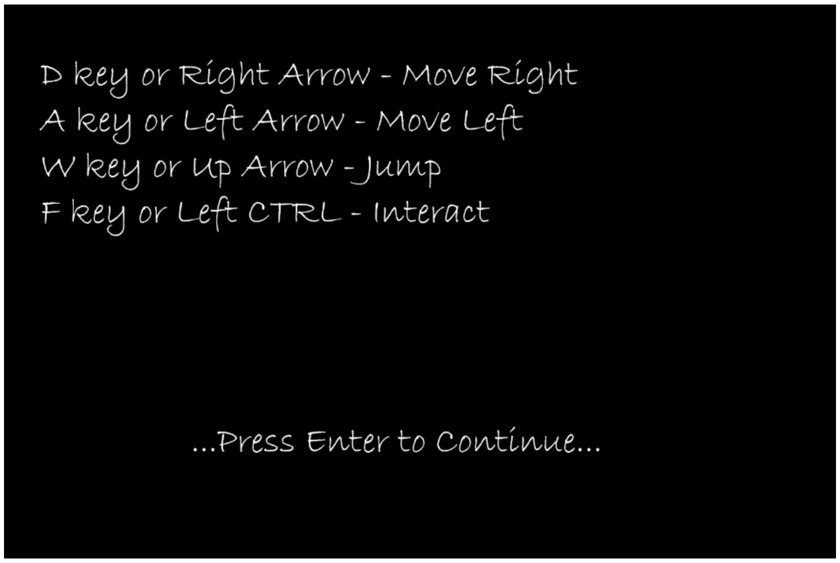
Main Menu

I decided the main menu would just be a simple screen: a pixelated Kizkalesi with the game's title and a simple button prompt. Unity maps the Enter button to the "Submit" Input Axis, so I decided that this would work just fine.



Controls Menu

Similar to the main menu, I simply wanted the controls menu to display the controls and continue to the game. I also decided that the enter button would continue to the game on this menu.

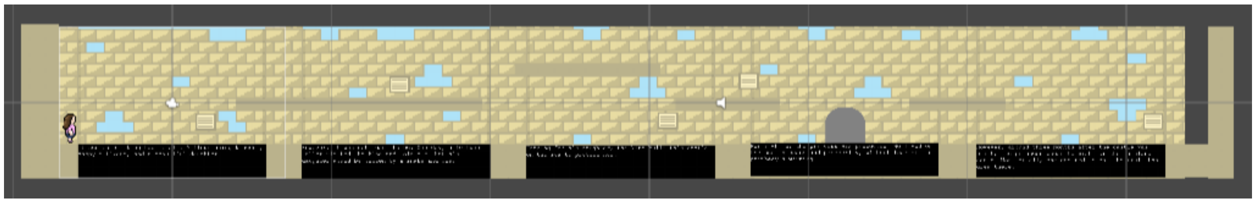


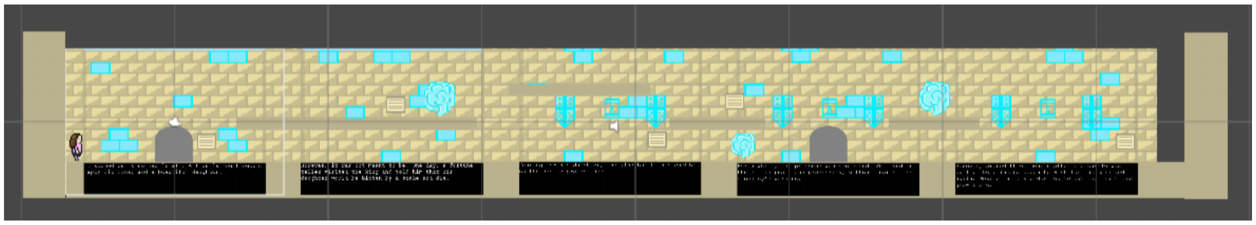
D key or Right Arrow - Move Right
A key or Left Arrow - Move Left
W key or Up Arrow - Jump
F key or Left CTRL - Interact

...Press Enter to Continue...

Downstairs Hallway

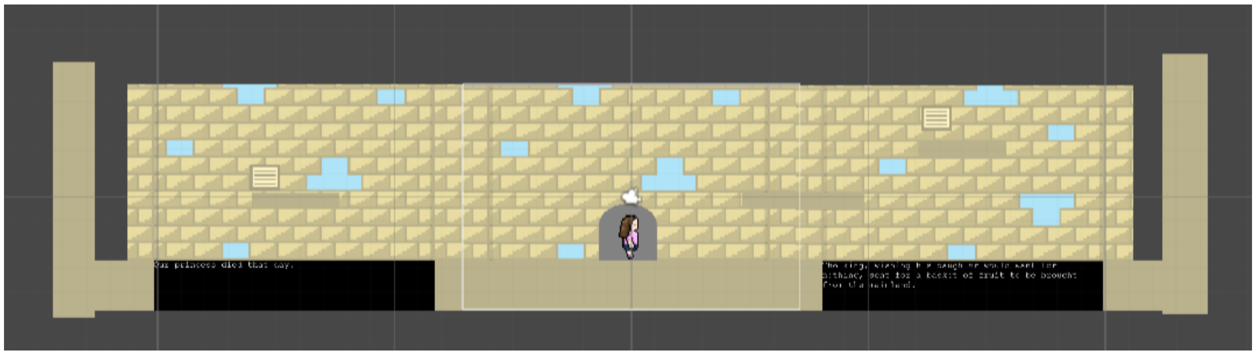
This is where the player will be most of the time, and I decided it would consist of two outdoor areas and then three hallways, one of which has a door going upstairs. Then off to the right there would be a way to go into the throne room. Since there were nine bits of lore I had made, I decided to put one lore tablet in each room, and then randomly placed platforms. I placed the dialogue over top of the floor and between the two pillars so it would be nicely aligned. I made two copies of this level, called “Explore” and “Ghosts,” and in the “Ghosts” level I randomly placed three ghosts that the player would have to avoid.



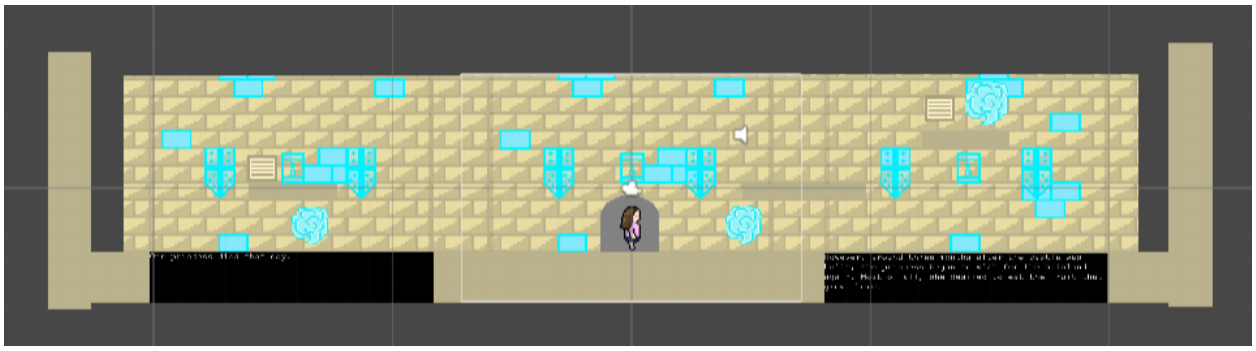


Upstairs Hallway

I decided I wanted to put the king’s and princess’ rooms on the second floor, so I decided to have a hallway going to each of these rooms. I then placed lore tablets in each hallway and placed platforms and ghosts similarly to how I placed them downstairs.


Our princess died that day.

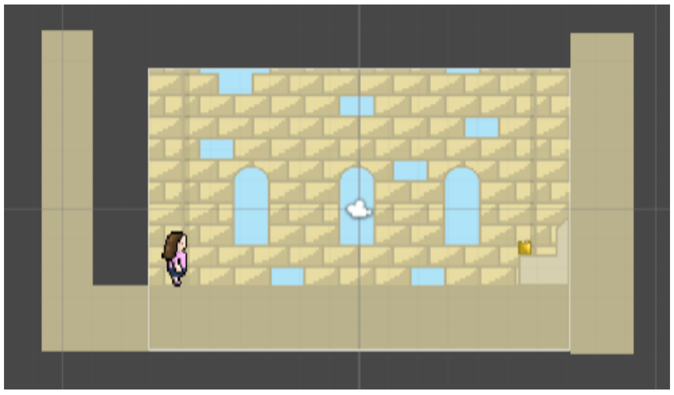
The king, wishing his daughter would want for nothing, sent for a basket of fruit to be brought from the mainland.

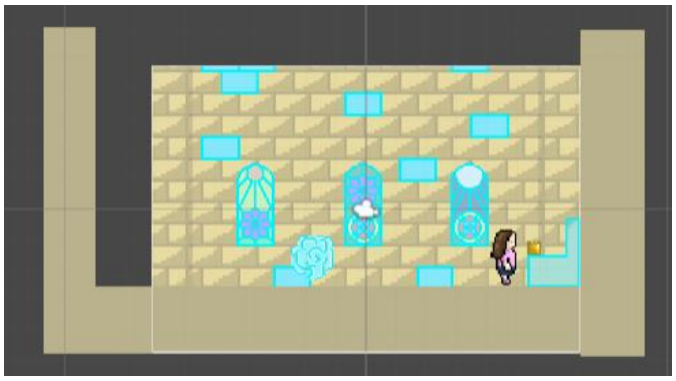

Our princess died that day.

[illegible]

Throne Room

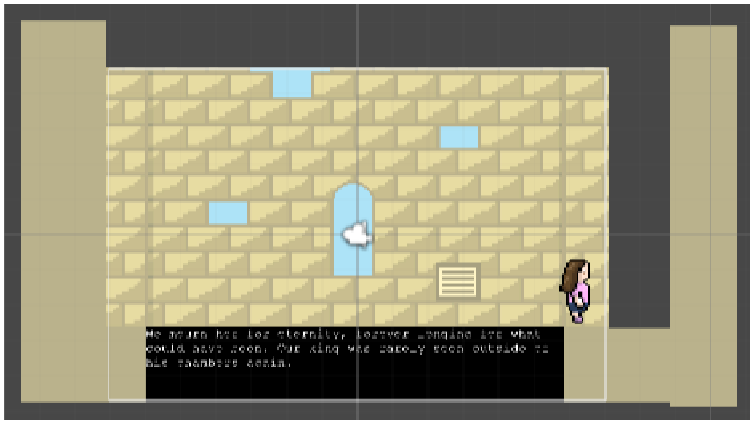
The throne room was only one room, and I was out of lore tablets, which I thought was okay because there was a different, just as important object to place: the throne with the king's crown on it. I didn't place any platforms in this room, but I did place one ghost in it for the ghost version to sort of teach the player the mechanic, since this would be the first time the player would be encountering a ghost.

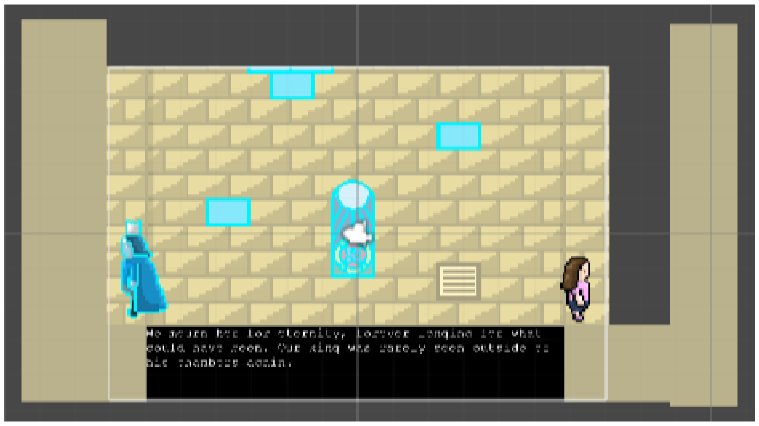




King’s Room

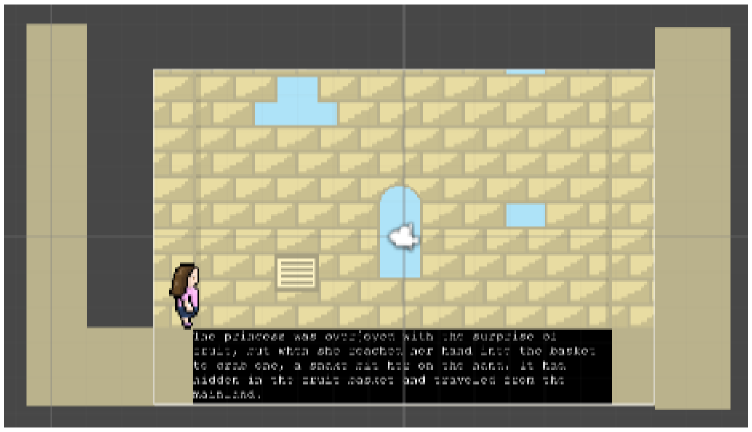
Similarly to the throne room, I didn’t want to place any platforms in the king’s room because I wanted the focus to be on the king when the player came to the room while the ghosts were summoned. I decided to place a lore tablet in the king’s room, the one that said that the citizens rarely saw him leave his chambers after his daughter died. I let this lore decision shape where I placed the rest of the lore later. In the ghost level I placed the king on the far left side of the room, turned away from the player. Like the other ghosts, I decided he could not be interacted with, forever mourning the loss of his daughter.


We mourn him for eternity, forever longing for what could have been. Our king was rarely seen outside of his chambers again.

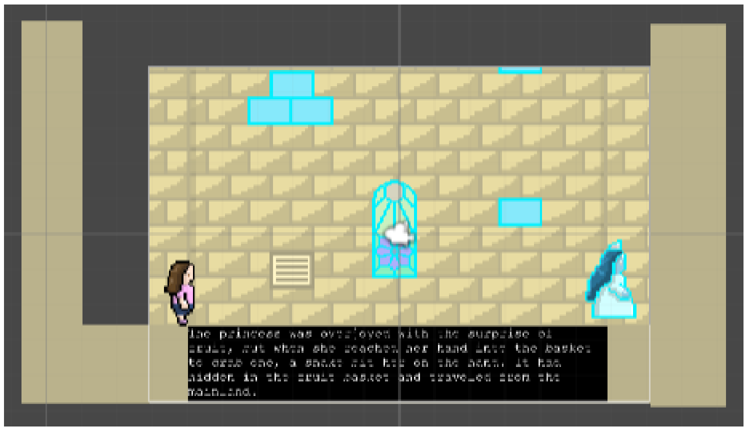

We mourn him for eternity, forever longing for what could have been. Our king was rarely seen outside of his chambers again.

Princess' Room

I built the princess' room almost exactly like the king's room, just flipped. The player comes in on the left, the lore tablet is a few steps to the right, and the princess is on the far right wall.



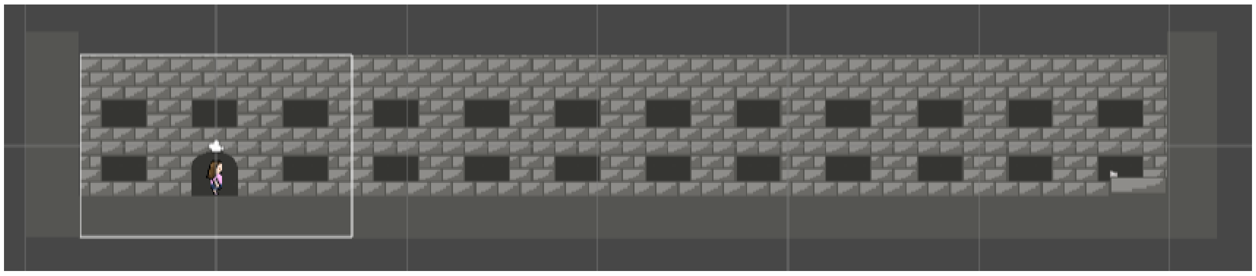
The princess was overjoyed with the surprise of fruit, but when she reached her hand into the basket to grab one, a snake bit her on the hand. It had hidden in the fruit basket and traveled from the mainland.



The princess was overjoyed with the surprise of fruit, but when she reached her hand into the basket to grab one, a snake bit her on the hand. It had hidden in the fruit basket and traveled from the mainland.

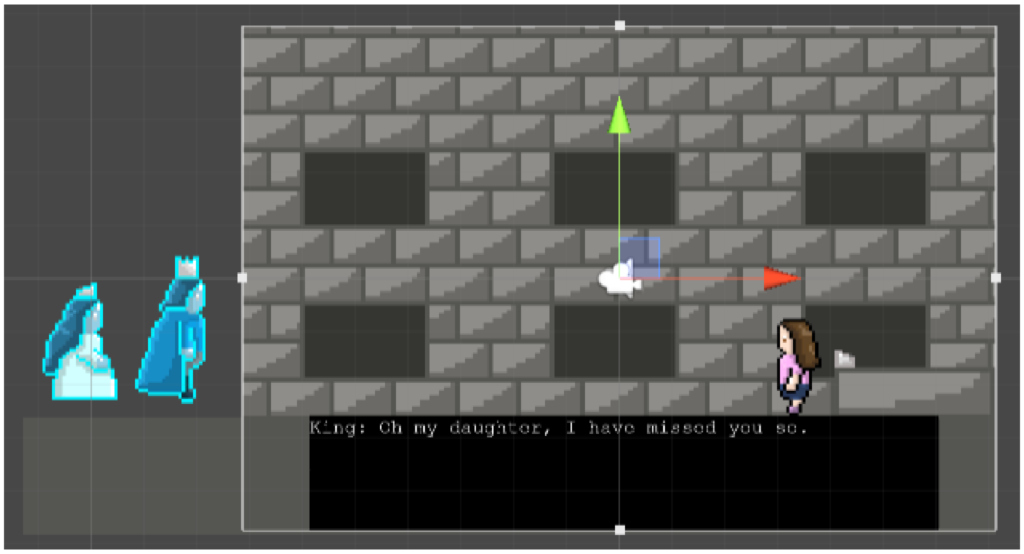
Crypt

I wanted the crypt to be slightly long and foreboding, much like long hallways in games like Undertale. So I decided to place the door at the far left, and then add three more rooms, with the princess' casket in the last room.



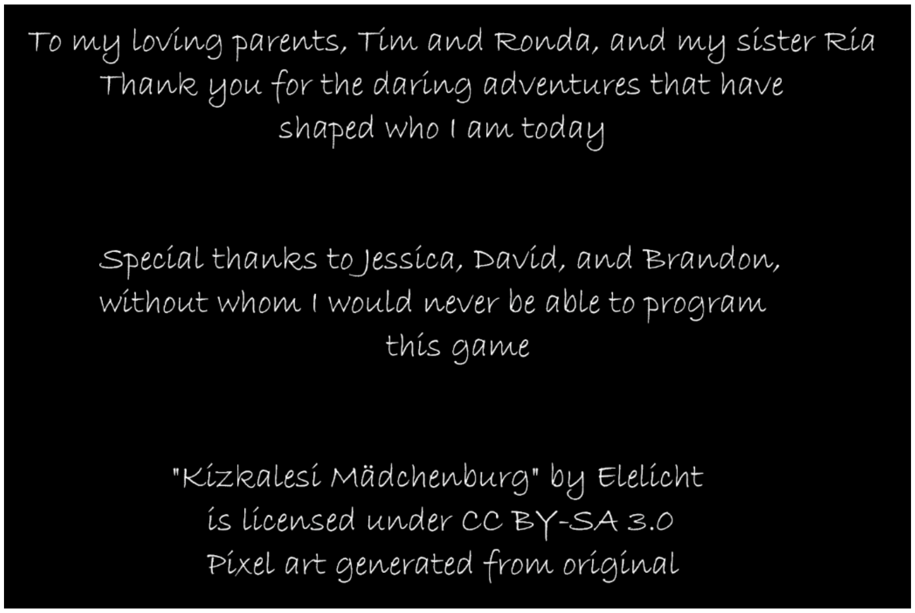
Cutscene

The cutscene is built very similarly to the last room of the crypt, but with no player or other room connections. Instead, the floor extends past the room on the left to keep the king and the princess when they are out of the cutscene. The dialogue is all in the same place on the bottom so the transition between them is seamless.


King: Oh my daughter, I have missed you so.

End Credits

I felt a little weird crediting myself in the end credits screen, so I decided instead to thank some of the people in my life and credit the photographer who took the Kizkalesi picture. My parents and sister traveled with me to many different historical landmarks, including Kizkalesi, and my friends taught me how to use Unity and cheered me on through the entire process so I had to shout them out. I decided to have the end credits loop back to the main menu, meaning I would want a windowed game rather than a full screen game so the player could exit whenever they wanted to.


To my loving parents, Tim and Ronda, and my sister Ria
Thank you for the daring adventures that have
shaped who I am today

Special thanks to Jessica, David, and Brandon,
without whom I would never be able to program
this game

"Kizkalesi Mädchenburg" by Elelicht
is licensed under CC BY-SA 3.0
Pixel art generated from original

Coding

While the code for this platformer was fairly simple, it was made somewhat more difficult by the fact that I had never coded in Unity before, and it had been a while since I had used C#. But with the help of a few of my friends and a lot of determination, I got through it just fine. The first thing I needed to figure out was how to control the player. I had the code continuously check for key presses, and then handle the physics when it happened.

Checking for key presses:

```
// check when the player has hit a movement key
 move = Input.GetAxis("Horizontal");

 // check when the player has hit a jump key
 if(notJumping)
     jump = Input.GetAxis("Vertical");

 // check and handle when the interact key is pressed (F key)
 interact = Input.GetAxis("Fire1");
```

Handling movement:

```
// handle movement
 // If move is 1, that means the player has pressed one of the keys to move right
 // If move is -1, that means the player has pressed one of the keys to move left
 // If move is 0, that means the player is stopped

 if (move == 1)
 {
     // set the animation to move right
     playerAnimation.SetBool("Right", true);
     playerAnimation.SetBool("Left", false);

     // get the player's position
     x = transform.position.x;
     y = transform.position.y;
     z = transform.position.z;

     // move the player right by a set interval
     transform.position = new Vector3(x+interval,y,z);
 }
 else if (move == -1)
 {
     // set the animation to move left
     playerAnimation.SetBool("Left", true);
     playerAnimation.SetBool("Right", false);

     // get the player's position
     x = transform.position.x;
     y = transform.position.y;
     z = transform.position.z;

     // move the player left by a set interval
     transform.position = new Vector3(x-interval,y,z);
 }
 else
 {
     // set the animation to stop moving
     playerAnimation.SetBool("Right", false);
     playerAnimation.SetBool("Left", false);
 }

 // handle jumping
 // If jump is equal to 1, that means the jump key has been pressed
 // Also check to make sure the player isn't moving when jumping so the player
doesn't go flying off the screen

 if (jump==1 && GetComponent<Rigidbody2D>().velocity.magnitude.Equals(0))
 {
     // set notJumping to false to stop the player from being able to jump twice
     notJumping = false;

     // set the jumping animation to start
     playerAnimation.SetBool("Jump", true);

     // add a force to the player to start the jump
     GetComponent<Rigidbody2D>().AddForce(new Vector2(0, 9), ForceMode2D.Impulse);

     // reset the jump variable
     jump = 0;
```

```
        }
```

At some point I accidentally sent the character flying into space from using too much force, but I corrected that eventually. One important note is that I implemented code to ensure the character couldn’t be sent flying into the air if the player pressed and held the jump button (like I do) and also made sure the player couldn’t jump unless it was on a block marked floor (the floor of the level and the platforms). I also put the code for handling movement and jumping in a FixedUpdate() function rather than the Update() function so the character will move the same speed no matter what.

The next biggest thing to handle was how to interact with objects. I already had the code for the key press, but needed code to figure out when the player was standing in front of the object to be interacted with. To do this, I used box colliders as triggers for when the player was in front of the object, and then Booleans to set whether the object could be interacted with or not.

```
private void OnTriggerEnter2D(Collider2D collision)
    {

        // if the player intersects a lore tablet, set tablet to true so the player can
       view the lore
        if (collision.gameObject.tag == "lore")
        {
            tablet = true;
        }

}

private void OnTriggerExit2D(Collider2D collision)
    {

        // when the player leaves a lore tablet, turn off any active dialogue and reset
the trigger
        if (collision.gameObject.tag == "lore")
        {
            // reset the trigger
            tablet = false;
            // hide the dialogue
            dialogue.SetActive(false);
        }

}
```

This was how I was able to get the player to activate lore, go through doors, etc. The other kind of trigger I used was one that activated when the player moved into and out of the trigger. Instead of creating a Boolean that would let the rest of the code know that things could be interacted with, this did something immediately. This was how I had the player walk between rooms and switched the camera.

```
private void OnTriggerExit2D(Collider2D collision)
    {

       // when the player leaves a camera pan trigger, move the camera a set distance
        if (collision.gameObject.tag == "pan camera")
        {
            // get the camera's current position
            x = camera.transform.position.x;
            y = camera.transform.position.y;
            z = camera.transform.position.z;

            // get the trigger's x value
            float xCollision = collision.gameObject.transform.position.x;

            // if the player has moved to the left of the trigger, move the camera left
            by a set size
            if(transform.position.x < xCollision)
            {
                camera.transform.position = new Vector3(xCollision - 7.01F, y, z);
            }
              // otherwise, if the player has moved to the right of the trigger, move the
              camera right by a set size
            else if(transform.position.x > xCollision)
            {
                camera.transform.position = new Vector3(xCollision + 7.01F, y, z);
            }
        }
    }


private void OnTriggerEnter2D(Collider2D collision)
    {

       // if the player intersects the throne room trigger, navigate to the throne room
```

```
        if (collision.gameObject.tag == "throne room")
        {
            // check to see if the ghosts have been summoned yet
            if (Globals.ghost)
                SceneManager.LoadScene(sceneName: "ThroneRoomGhosts");
            else
                SceneManager.LoadScene(sceneName: "ThroneRoom");
        }

       }
```

The last thing I implemented in the player code was global variables. I know that global variables are not usually good practice, but every time a new level was loaded, a new instance of the player was created and all the values that the old player instance had were gone. If there was something that the new instance needed to know, a global variable would be the easiest way to let it know. The five instances where global variables were needed are with ghosts, the throne room, the princess' room, the king's room, and going downstairs. The player needed a global variable to see when the levels had switched to ghosts, and these rooms were in places that the new player instance had to adjust the position of the player sprite and the camera because it was away from the normal starting position. Neither of these instances could be avoided, so I used the global variables.

```
public static class Globals
{
    public static bool ghost = false;
    public static bool door;
    public static bool princess;
    public static bool king;
    public static bool throneRoom;
}



void Start()
    {

       // check the global variables

        // if the door is active, then that means that the player has moved from the
       upstairs to the downstairs and needs to be repositioned
        if(Globals.door)
        {
            transform.position = new Vector3(42.3F, -1.6F, 0F);
            camera.transform.position = new Vector3(42.3F, 0F, -10F);
            Globals.door = false;
            Debug.Log("Door");
        }

       }
```

```
private void OnTriggerEnter2D(Collider2D collision)
    {

       // if the player intersects the throne room trigger, navigate to the throne room
        if (collision.gameObject.tag == "throne room")
        {
            // check to see if the ghosts have been summoned yet
            if (Globals.ghost)
                SceneManager.LoadScene(sceneName: "ThroneRoomGhosts");
            else
                SceneManager.LoadScene(sceneName: "ThroneRoom");
        }

       }
```

From there, everything else was simply switching between levels. The Main Menu, End Credits, and Controls Menu had the same two lines of code pretty much.

```
void Update()
    {
        // check to see if the enter button has been pressed to start the game

        float start = Input.GetAxis("Submit");

        if(start != 0)
        {
            SceneManager.LoadScene(sceneName: "Explore");
        }
    }
```

The cutscene was slightly different because it had animations in it, but once I looked up how to get that into my code, it was fairly simple, too.

```
void Update()
    {
        // check to see if the cutscene is done to go to the end credits
        if(director.state != PlayState.Playing)
        {
            SceneManager.LoadScene(sceneName: "End Credits");
        }
    }
```

And with that, the code was done.

Cutscene Animation

I had dabbled in animation before, but never at this level and never using Unity. However, after looking it up, it was fairly easy to pick up. I wanted a very simple scene where the king reunites with his daughter, they are freed, and they go to the afterlife together. Thus, I started by creating dialogue.

?: You!

King: That is my daughter's crown! What are you doing with it?

?: Wait!

King: Could it be…? I have been trapped alone in my chambers for so long, I fear I may be imagining her voice…

Princess: I have been trapped as well, Father. But this young woman has set us free! Do not harm her.

King: Oh my daughter, I have missed you so.

Princess: As have I. Let us go, and never be apart again.

King: Thank you for returning my daughter to me. We are forever grateful to you.

From there, I imagined what I wanted the characters to be doing when each of these lines was spoken. First, I wanted the player character to be looking at the coffin, then turn when the king says, “you!” I didn’t want the player character to do anything after that really. Then the king would begin to chase after the player, angry that they disturbed his daughter’s crown, when the princess suddenly comes in to stop him. The king turns, then when his daughter comes near, he begins walking towards her, pausing for a moment before rushing to her. At this point, his original appearance is restored to him as he is reunited with his daughter. Then the princess regains her original appearance too. They turn and begin to head out, but the king pauses for a moment, turns, thanks the player, then leaves.

Final Product

I am extremely proud of the game I was able to create in the last three weeks. It has a story that means something to me and brings a sad story to a happy completion, and I was able to learn how to use Unity for the first time.

The most important lesson I learned on this project are that art for games take an extremely long amount of time. Since my sister is a fine arts major, I did know that art took time, but now that I have gone through the process of generating art for a game myself, I have a newfound respect for artists and animators who do this for a living.

Another thing I learned is that Unity is extremely easy to learn. While it all seemed overwhelming at the beginning, and I had some issues building my project into an .exe file, most of the time I could either look up my problems on Google or ask one of my friends, who would easily explain it. Some problems even somehow fixed themselves.

Overall, I am really glad I followed that graduate student's suggestion to build a game from the ground up. Not only do I have a portfolio to deliver that I am truly proud of, but I feel like I learned a lot about the game development process overall.

A demonstration of my game can be found here:

https://www.youtube.com/watch?v=bykJHlQlQ9g