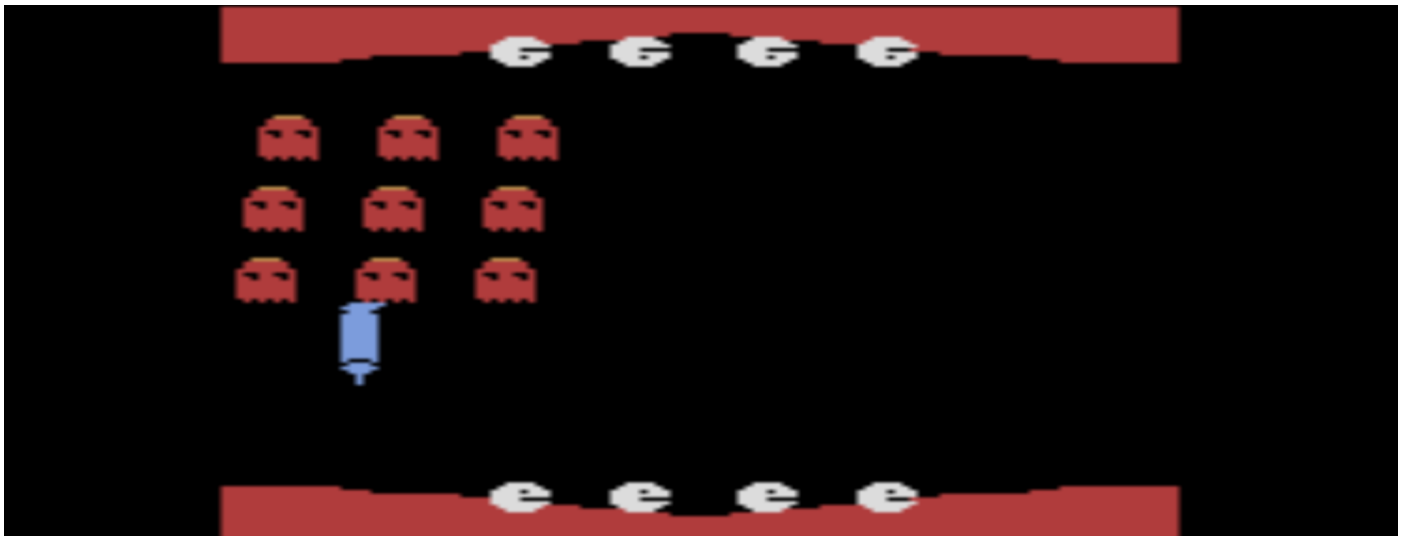
The Pac-Manification of Plaque Attack for the Atari 2600



For our second major project in the Fall Semester 2023 Game Systems class at Swarthmore College, we were tasked with hacking an Atari 2600 game and making whatever alterations we saw fit. Initially planning to alter *Pac-man*, we were unsure how we wanted to change *Plaque Attack.* We brainstormed many ideas before settling on the funniest one.

Contents

History of *Plaque Attack*

There is very little record of *Plaque Attack* on the internet to examine. That which we do have mostly cites the same Wikipedia page. Despite this we have pieced together a rough backstory to this dental themed romp.

Plaque Attack was released in 1983 by Activision for the Atari 2600. The game was developed by Steve Cartwright, who also worked on Barnstorming, Megamania, and several other games. The concept was that the game was intended to be a pro-social force promoting good dental hygiene. This likely would have been more successful if the game had found wide

commercial success. Publications at the time rated the game around a B/B+ and remarked that while it was a fine game, there was nothing particularly special about another side shooter.

The largest impact the game seems to have had is a partnership with the toothpaste brand Aquafresh that cross promoted both products. Perhaps the fact that a similar game called “Tooth Invaders” had come out two years before meant that the game was doomed before it was released.

The Game

The game initializes with 8 teeth and a blank background. The player character is a tube of toothpaste that spawns near the center of the screen. The game begins when the player moves the toothpaste, at which point a number of junk foods spawn in and begin moving towards the teeth.

The core game loop functions as follows;

- A number of junk foods spawn in and start moving towards the teeth
- If you hit the junk food with a blast of toothpaste, it disappears
    - If a food hits the tooth it decays
    - If you destroy all junk food you progress to a new level
- If you run out of toothpaste you can no longer shoot

The loop continues until all teeth decay, At which point the game ends. Additionally, the player earns points for every junk food they shoot. Upon scoring 200 points, the player earns an additional tooth back. Also, the sprite for the junk food changes every level.

Our Changes

All of our changes to *Plaque Attack* were aesthetic in nature. We altered the shape of every junk food sprite as well as the tooth sprite, and we changed the colors of the junk food sprites. Each change is elaborated below.

Plaque Attack	Pac Attack	Description

```
HamburgerGraphics
    .byte $00 ; |........|
    .byte $FF ; |XXXXXXXX|
    .byte $FF ; |XXXXXXXX|
    .byte $7E ; |.XXXXXX.|
    .byte $FF ; |XXXXXXXX|
    .byte $7E ; |.XXXXXX.|
    .byte $FF ; |XXXXXXXX|
    .byte $7E ; |.XXXXXX.|
    .byte $FF ; |XXXXXXXX|
    .byte $FF ; |XXXXXXXX|
    .byte $7E ; |.XXXXXX.|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
```

```
HamburgerGraphics
    .byte $00 ; |........|
    .byte #%01010101;$44 . |.X.X.X.X|
    .byte #%11111111;$44 . |XXXXXXXX|
    .byte #%11111111;$44 . |XXXXXXXX|
    .byte #%11111111;$44 . |XXXXXXXX|
    .byte #%11111111;$44 . |XXXXXXXX|
    .byte #%11111111;$44 . |XXXXXXXX|
    .byte #%11011101;$44 . |XX.XXX.X|
    .byte #%10011001;$44 . |X..XX..X|
    .byte #%11111111;$44 . |XXXXXXXX|
    .byte #%01111110;$44 . |.XXXXXX.|
    .byte #%01111110;$44 . |.XXXXXX.|
    .byte #%00111100;$44 . |..XXXX..|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
    .byte $00 ; |........|
```

Using the HamburgerGraphics as our example junk food, we modified the individual bytes to create the shape of a ghost. The sprite, as seen here, is clearly upside-down so we took that into account while designing. However, for some odd reason the sprite still had a stripe of a different color on the top-most non-zero byte of the new model. We could not figure out why that is, but we thought it was cute so we left it as such.

```
ToothGraphic
    .byte $00 ; |........|
    .byte $42 ; |.X....X.|
    .byte $42 ; |.X....X.|
    .byte $E7 ; |XXX..XXX|
    .byte $E7 ; |XXX..XXX|
    .byte $FF ; |XXXXXXXX|
    .byte $FF ; |XXXXXXXX|
    .byte $FF ; |XXXXXXXX|
    .byte $FF ; |XXXXXXXX|
    .byte $FF ; |XXXXXXXX|
    .byte $FF ; |XXXXXXXX|
    .byte $FF ; |XXXXXXXX|
    .byte $C3 ; |XX....XX|
```

```
ToothGraphic
    .byte $00 ; |........|
    .byte $00 ; |.X....X.|
    .byte $00 ; |.X....X.|
    .byte $00 ; |XXX..XXX|
    .byte $00 ; |XXX..XXX|
    .byte #%00111100;$1E
    .byte #%01111110;$1E
    .byte #%11111111;$1E
    .byte #%11110000;$1E
    .byte #%11111111;$1E
    .byte #%11110111;$1E
    .byte #%01111110;$1E
    .byte #%00111100;$1E
    .byte $00 ; |XX....XX|
```

The ToothGraphic is what must be protected, what is on the gums of the screen, and what we turned into Pac-Man. We made a white Pac-Man that looks a little silly, and it is a generic design. This sprite is used for each tooth, meaning it is placed upside-down and mirrored for the top row of teeth, and even the Lives counter in the top left. Oddly, the stripe of a different color did not appear on this sprite.

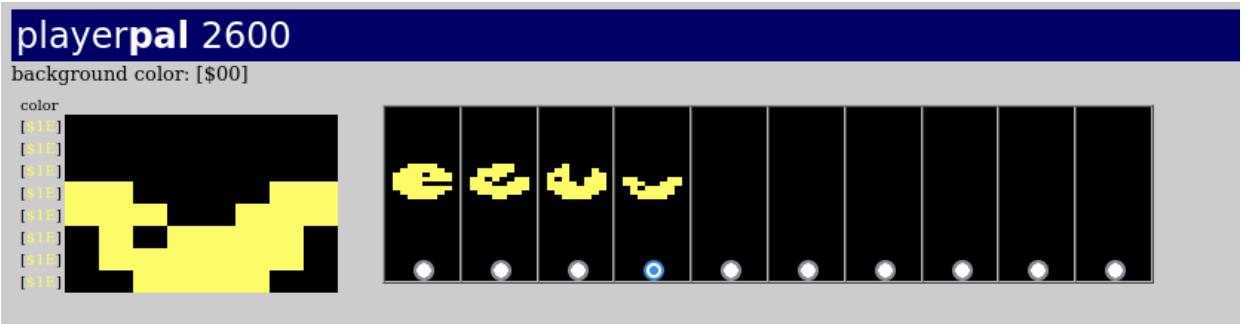
```
JunkFoodColors
HamburgerColors
   .byte BLACK, RED_ORANGE + 8, RED_ORANGE + 8, YELLOW + 2, GREEN + 4, RED + 2
   .byte YELLOW + 10, YELLOW + 2, RED_ORANGE + 8, RED_ORANGE + 6, RED_ORANGE + 4
StrawberriesColors
   .byte BLACK, RED + 2, RED + 2, RED + 4, RED + 4, RED + 4, RED + 4, RED + 4
   .byte RED + 2, RED + 2, GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6
HotdogColors
   .byte BLACK, YELLOW + 4, YELLOW + 6, YELLOW + 8, COLOR_HOT_DOG + 2
   .byte COLOR_HOT_DOG + 2, YELLOW + 8, YELLOW + 4, BLACK
GumDropsColors
   .byte BLACK, RED + 2, RED + 4, RED + 4, RED + 4, RED + 4, RED + 4, RED + 2
   .byte GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6
FriesColors
   .byte BLACK, WHITE - 2, WHITE - 2, WHITE - 2, WHITE - 2, WHITE - 2, WHITE - 2
   .byte YELLOW + 4, YELLOW + 8, YELLOW + 8, YELLOW + 8, YELLOW + 8, YELLOW + 8
DonutsColors
   .byte BLACK, RED_ORANGE + 4, RED_ORANGE + 4, RED_ORANGE + 4, RED + 10
   .byte RED + 10, RED + 10, RED + 10, RED + 10
CandyCanesColors
   .byte BLACK, WHITE - 2, WHITE - 2, RED + 4, RED + 4, WHITE - 2, WHITE - 2
   .byte RED + 4, RED + 4, WHITE - 2, WHITE - 2, RED + 4, RED + 4, WHITE - 2
IceCreamConesColors
   .byte BLACK, RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8
   .byte RED_ORANGE + 8, RED_ORANGE + 8, GREEN + 6, GREEN + 6, GREEN + 6
   .byte WHITE - 2, RED + 4, RED + 4, RED + 4
```

```
JunkFoodColors
HamburgerColors
   .byte RED + 4, RED + 4, RED + 4, RED + 4, RED + 4, RED + 4
   .byte RED + 4, RED + 4, RED + 4, RED + 4, RED + 4, RED + 4
StrawberriesColors
   .byte RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8
   .byte RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8, RED_ORANGE + 8
HotdogColors
   .byte GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6
   .byte GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6, GREEN + 6
GumDropsColors
   .byte BLUE + 4, BLUE + 4, BLUE + 4, BLUE + 4, BLUE + 4, BLUE + 4
   .byte BLUE + 4, BLUE + 4, BLUE + 4, BLUE + 4, BLUE + 4, BLUE + 4
FriesColors
   .byte PURPLE, PURPLE, PURPLE, PURPLE, PURPLE, PURPLE
   .byte PURPLE, PURPLE, PURPLE, PURPLE, PURPLE, PURPLE
DonutsColors
   .byte RED, RED, RED, RED, RED, RED
   .byte RED, RED, RED, RED, RED, RED
CandyCanesColors
   .byte PURPLE - 6, PURPLE - 6, PURPLE - 6, PURPLE - 6, PURPLE - 6, PURPLE - 6
   .byte PURPLE - 6, PURPLE - 6, PURPLE - 6, PURPLE - 6, PURPLE - 6, PURPLE - 6
IceCreamConesColors
   .byte GREEN + 1, GREEN + 1, GREEN + 1, GREEN + 1, GREEN + 1, GREEN + 1
   .byte GREEN + 1, GREEN + 1, GREEN + 1, GREEN + 1, GREEN + 1, GREEN + 1
```

The JunkFoodColors were also changed, and we changed them all individually. Since we were not mixing colors within the sprites, the colors in the code look significantly cleaner. Each byte could hold the information per bit in the byte, but because of how our ghosts were designed we only used 6.

Regarding the “junk food” modification, it was necessary to change each individual sprite option (i.e. the hamburgers, the strawberries, the hotdog, etc.) because of how the sprite is called into play. Originally, we had intended to change all of the “junkfood” calls to immediately point towards the “hamburger” sprite, but each individual sprite was used within the gameplay as part of the “junkfood” categories, and we decided it would be easier to make all of the sprites individually the same design but have armies of different colored ghosts instead. The “junkfood” label is essentially the equivalent of a parent class, and each individual food item a child class of it, and it is nearly impossible to find where the child classes are individually noted within the gameplay.

Moreover, you will notice that we did not add the nice comment on the right that shows the displayed bitmap for each of the sprites. That is because we found a very handy tool through 8bitworkshop that helped give us the map for each sprite, and we had to manually type out the one ghost display, which we did not think was necessary for every design-based change. The website “playerpal2600” is directly linked on the 8bitworkshop page, under the Tools→Atari2600 label in the hamburger menu, and allows us to both import and export bitmaps, but the comments on the side actually stop the program from working properly. We used this tool to make a lot of our sprite changes as it helped see the visual for multi-framed sprites.



Pac-Man as teeth was already a given considering the ghosts, but the fact that the teeth decay made it a much more entertaining experience. The tooth decay has multiple frames to create the animation, which means we could try to make the original Pac-Man death animation, so naturally we tried our best to mimic it. The image above shows the best frame-by-frame of the animation, as it was taken during the design process. When Pac-Tooth is hit and starts to decay, it turns yellow like Pac-Man should be, but then immediately collapses back as its decay animation. Probably the best part is that, because of the racing the beam aspect, when this animation reaches the last frame the visual will look like a full decay-cycle from top to bottom of the Pac-Tooth.

As a detail, as we were finishing this write-up we realized that it would take very little time to change a minor detail. We replaced the bitmaps of “Activision” and “Copyright” to say PacAttack and our names. There was no reason to do so but it seemed like a nice little modification.

Why?

After deciding to hack this game we knew we had to change something. With the words “Plaque” and “Pac” rhyming, we thought it would be a fun change to see if we could somehow mimic as much of *Pac-Man* as we could within *Plaque Attack*. Taking inspiration from how the ghosts pursue Pac-Man in the titular game, we wanted to change the junk food into ghosts and the teeth into *Pac-man* to mimic that dynamic. So, our final product is both a pun and a coherent, functioning game.

Conclusion

Although none of our changes altered game mechanics, just by altering the sprites we were able to completely change the experience of playing *Plaque Attack.* This lab was a good exercise in understanding the assembly code behind the games we have been looking at and really grasp the constraints of early game designers. We also gained a newfound appreciation for how direct of a role binary code played in sprite graphics. We will never think about the phrase “8-bit” the same way.