JOUST

Published by Williams Electronics, released for arcades in 1982. Also published by Atari, Inc. for several Atari versions, Apple II, and Max. HAL Laboratory also ported Joust to the NES.

* <https://www.free80sarcade.com/nes_Joust.php>
* <https://en.wikipedia.org/wiki/Joust_(video_game)>

Joust is a game where you play as a knight armed with a lance riding an ostrich whose goal is to defeat the enemy knights riding around on buzzards. Players and enemies can flap to fly around the screen and to various platforms. When an enemy and player collide, whichever has the high ground gets the kill.

**Accomplished Scope**

* Flying mechanic – flap to fly
* The original 3 enemy jousters
  + Bounder, Hunter, Shadow Lord
* “Bouncing” when collision with altitude tie (no kill)
  + Note that this had a very tiny window (had to be exact, should have had a larger tie window)
* Player lives – start with 5 (up from 3 in design doc, 5 is the original version’s start number)
* Waves – level progression
  + Added original first 30 waves, along with repeating wave afterwards (this is up from the design doc’s planned 3 waves)
* Basic main menu and game start/end loop
  + Just press ‘enter’ to start game
  + Added end game screen to display player’s final score
* Sound effects

**Accomplished Stretch Goals**

* Wall wrapping (not just go off screen and move to other side, but pixel wall wrapping as you go off screen)
* Wave based progression up to endless mode

**Implementation** – Time is in hh:mm

MS1: Character moving in 4 directions (no flapping)

* Time Estimation: 16 hours
* Completion date estimation: 11/16
* Details: Initial project setup, asset gathering, initial character (no animations), input system. Character will be able to move left/right/up/down at first
* Actual time: 4:51
* Actual completion date: 11/16
* Final Commit Number: 37252
* Notes: Initial project setup and character movement wasn’t challenging, just took time to get everything setup and become comfortable with framework. Movement did NOT include flying/flapping as no physics or gravity had been implemented yet.

MS2: Add the first enemy & collision system

* Time Estimation: 8 hours
* Completion date estimation: 11/18
* Details: Add enemy with basic enemy system that is extensible. Setup collision detection and reaction system.
  + Note for killing: killing entity’s center must be above the other enemy’s center and lance must be colliding. Killing an enemy should give the player points as well.
* Actual time: 20:23
* Actual completion date: 11/29 / (Missing file addition): 11/30
* Final Commit Number: 39197 / (Missing file addition): 39263
* Notes: Collision system was more challenging than expected. Not detection, but how to react to collision. Needed to adjust reactions throughout project once the initial system was in place. Could have definitely been architected much better, but once the desired effects were achieved, I moved on to other more pressing issues.

MS3: Flying/Gravity system on player/enemies

* Time Estimation: 12 hours
* Completion date estimation: 11/21
* Details: Add gravity to players and enemies. Give player the ability to flap wings (no animation yet) to fly.
* Actual time: 3:56
* Actual completion date: 11/19 / (Minor change with input keys): 12/1
* Final Commit Number: 37822 / (Minor change with input keys): 39576
* Notes: Was pretty reasonable and not too difficult to implement. Could have been tuned more (flapping and terminal velocities). Terminal velocities were stored directly in the entities that were affected by physics so changing and tuning was able to happen outside of where the physics were being applied.

MS4: Enemy AI

* Time Estimation: 20 hours
* Completion date estimation: 11/23
* Details: Give enemy “random” movement and flying to move around screen/map. Move out common functionality of player & enemies to the entity class so both can call said functionality.
* Actual time: 5:13
* Actual completion date: 11/30
* Final Commit Number: 39351
* Notes: Deciding how enemies should behave was pretty difficult. The actual implementing was reasonable to accomplish. Should have given the AI more information such as where platforms were and potentially where the lava could have been. If that information was given to them, it would have been much more realistic to get the lava feature into the game.

MS5: Layout the map and setup waves

* Time Estimation: 10 hours
* Completion date estimation: 11/25
* Details: Layout map with sprites that have colliders to act as walls/platforms. Set player respawn locations. Setup a system to create and run waves with enemy spawning. Should allow specific enemies to be added along with their spawn locations.
* Actual time: 11:51
* Actual completion date: 11/30
* Final Commit Number: 39436
* Notes: I used invisible collision boxes located around the image of the platforms on the map to accomplish the illusion of platforms. If I had isolated the sprites of the platforms, combined them with a collision box, and placed them at their appropriate locations, it would have been relatively easy to accomplish platforms turning off/on based on the current wave.

MS6: Give player lives

* Time Estimation: 4 hours
* Completion date estimation: 11/26
* Details: Give player a set number of lives at the beginning of game. Add player respawning after death. End game when player is out of lives.
* Actual time: 0:42
* Actual completion date: 11/29
* Final Commit Number: 39115
* Notes: This MS was very easy to implement as throughout the development of the player and UI looping, I left spots to put in player lives. Originally, I had collision with players/enemies destroying the objects themselves for collision testing. Eventually I swapped that to just disabling them, and it made moving them to their respawns and enabling them much easier. The harder part of this MS was getting the end game state as I had to change levels when that happened. This pushed me to work on MS 6 and MS 7 in tandem for a little bit.

MS7: UI Pass

* Time Estimation: 16 hours
* Completion date estimation: 11/29
* Details: Setup and add UI to game loop, such as score and lives display. Anything not changing at all will be part of the background image itself. Add Main Menu (image w/ “press <button> to start” and a quit option). End game screen should show user their score and allow them to loop back to main menu.
* Actual time: 14:41
* Actual completion date: 12/1
* Final Commit Number: 39575
* Notes: This was a difficult MS all around. Getting the game to loop between levels was reasonable once I was able to define levels in a meaningful way. Though UI took a while for me to get implemented and looking “good” to me.

MS8: Animation

* Time Estimation: Unknown / As much as possible with remaining time
* Completion date estimation: 12/2
* Details: This includes all animations. Player/enemy running, slowing down, flapping.
* Actual time: 13:23
* Actual completion date: 11/30
* Final Commit Number: 39367
* Notes: Animations were hard for me to conceptualize, so a lot of thinking went into this MS. Once I got a basic animation sequence working, I left the artifacts that were being drawn until the last few days where I cleaned them up. Another difficult part for me was putting the animation transitions in the right location in code. I probably should have had a “state” that the player/enemy were in rather than attaching references to animations to them.

MS8+: Polish / Stretch Goals

* Time Estimation: Unknown / As much as possible with remaining time
* Completion date estimation: 12/2
* Details: This includes animations as well as anything in the “Planned Scope” section that has been marked as ‘Stretch’ in addition to anything in the “Omissions / Stretch Goals” section. The order of these goals will be decided upon reaching or getting close to this shippable assuming there is time to complete any of them.
* Actual time: 8:23
* Actual completion date: 12/1
* Final Commit Number: 39683
* Notes: I wanted to have a hiscores screen / leaderboard at the end of my game. However, when I realized there was not enough time for me to put in to make it as good as I wanted, I pivoted to just a screen that showed the player’s score and the title of the game.

Cleanup:

* Time Estimation: Unknown / As much as possible with remaining time
* Completion date estimation: 12/2
* Details: This entails any code cleanup or refactoring. Can occur throughout the development process or at the end.
* Actual time: 11:53
* Actual completion date: 12/1
* Final Commit Number: 39635
* Notes: I did a good amount of code cleaning throughout the development process. However, I am not happy with the inheritance structure of several of my objects as well as the names of several classes. If I had more time I would certainly go back and refactor this code more.

**Risks and Mitigations**

* Time/Scope (72 planned hours)
  + This risk is most certainly the scariest for me. Considering I spent ~98 hours on the Assembly Game and the estimates were much lower, I expect my planned hours on this game to take longer than my current estimates.
  + I setup the schedule to get a basic version of the game with minimal mechanics earlyish on in case other tasks take much longer.
  + Another risk here is that I have personal obligations over the duration of this assignment as one of my sisters is in town as well as celebrating both of my sisters’ birthdays. I gave padding in the schedule during those dates as well set my “completion date” for the shippable game a weekend before the showcase. The extra time can be spent adding polish and stretch goals or bug fixing and finishing the final implementation.
  + For several of the features I planned to add, in the “Planned Scope” section I added a mitigation version in case I’m running out of time to meet the original goal.
  + All of the above mitigations were used on this risk. Clearly my time estimates are still below their actuals, however I believe I’m getting much closer than I was for the Assembly Midterm Game. This was a completely expected risk that I believe will happen on most projects I do. My final time contributed to this project was 95 hour and 17 minutes time tracked. Honestly, even when I finished my minimum MS, I felt the need/want to continue to polish and add more features. So as long as I meet the minimum, I set myself I am happy. Note that this does not mean I will set a lower bar for myself in the future, I will still continue to set a high bar and do my best to surpass it.
* Assets
  + Finding sprites and clipping them is a weakness of mine, specifically clipping them to use in game. To help mitigate this I’ve saved and am using the site: <https://www.spriters-resource.com/search/?q=Joust>
  + While I’ve found player sprite sheets and sprites for enemies and platforms, I may need more for the UI.
  + If I can’t find more assets I need, I’ll play the emulator and use screenshots to gather the remaining ones I need.
  + A mitigation here would be to use basic shapes if I can’t get certain assets, or potentially make my own main menu / quit game screen.
  + Clipping the art was most certainly still a weakness, I’m still not sure where to put the data (pixel locations on the sprite sheet) in code to be as organized, modular, and general as possible. I was originally using a sprite sheet that did not contain most of the sprites I needed for UI and the game map, but did end up finding another on the website linked above that I pivoted to that had just about everything I needed. I had to create a few sprites for the enemies, but that did not take too much time away from other MS. I do think that my mitigation to use basic shapes would have been a very bad mitigation here as art and animations matter tremendously.
* Knowledge of Game Framework
  + Prior to writing any lines of code, I need to spend time to read through the provided game framework and understand what is available to me and how to use it.
  + This time is NOT included in my Time/Scope estimation.
  + A mitigation to this is to dedicate time every day starting now towards reading through it. Time between assignments, when walking home, watching tv, time prior to bed, etc. Pretty much find small times to read through sections of it to gain a full understanding of what all the code is doing.
  + I did use the mitigation here and reading through the framework thoroughly provided me with a very good base knowledge of how to use it to begin working on Joust. I did use a good amount of the prior game code such as the level manager and object manager that was given to use so I could focus on other gameplay related MS.

**Post-Mortem**

**Positives**

* I think the character movement and feel of the controls came out as nice as I wanted. It felt good to play and move around the screen to avoid and kill the enemies as they swarmed you.
* I also believe that even though I didn’t get to a hiscores screen, my replacement of displaying the player’s score and ‘lerping’ it from it position in the game screen to the middle as well as ‘lerping’ the Joust title from above offscreen looked great and was a reasonable way to “end” the game. That allowed me to accomplish a reasonable game loop that I was struggling to conceive of without hiscores.
* My personal organization and work/life balance was much more reasonable for this project than it was for the Assembly project. Even though I put more time into this project and in a shorter time period, I had a better home life and mental state and came out of this project with something that I am proud of.

**Negatives**

* Players and enemies were too small relative to the map. I thought that this would provide the player a bit more room to fly around the map dodging and killing the enemies, however it was an immediate note of difference to any previous Joust enthusiasts.
* Players and enemies were also much too slow relative to the original game. I struggled personally to get the speeds higher and properly test the game, but I recognize that’s just a skill diff for me and I should have played more Joust (at least on an emulator) to get good enough to have the game running at full speed. Though note here that I left it pretty changeable, as I can edit the code where terminal velocity is set for each entity very easily and within 5 minutes.
* My collision system was not setup in a very organized way and I am disappointed in myself for that. Detecting collision was reasonable for me as I had done it before, however responding to the collision in an appropriate manner was much more difficult for me to accomplish. This system was tangled throughout the entities that were collidable similar to weeds in a poorly manicured lawn.
* I am not happy with my inheritance hierarchy and naming of several classes throughout the project. I had several misnomer classes and identical variables in different classes that inherited from the same parent, and also classes that inherited from a parent that did not need parts of said parent.

**Learnings and Key Takeaways**

The biggest key takeaway I have from this project is pre-planning object and inheritance structures prior to beginning work that requires them. Several times I created classes that were needed immediately, with little forethought about the future needs/children of the class. A few times this resulted in a necessity to refactor classes before moving on with the project, slowing down development and have a less organized inheritance structure. I lost several hours to refactoring when I needed to move to more important systems and game features. I anticipate this learning experience will give me more foresight into planning inheritance and object hierarchies. I liken this to learning not to tough the stove while its hot after burning your hand on it once (or twice, or several times).