Balancing Act: An Interactive Tool for Fabricating Calder-Style Hanging Mobiles

Liane Makatura*
Dartmouth College

Catherine Most†
Dartmouth College

Gloria Li‡
Dartmouth College

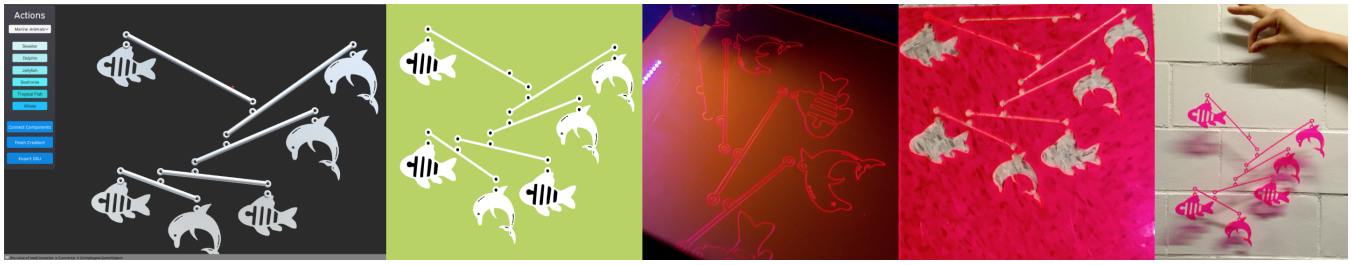


Figure 1: *An overview of our pipeline, showing (a) our mobile-designing UI, (b) the outputted SVG ready for (c) laser cutting. Then, the user is able to (d) punch out the pieces and (e) assemble their custom mobile using jump rings and a clear monofilament for suspension.*

Abstract

We present a novel end-to-end system that allows users to easily design and fabricate custom mobiles, mimicking the style of those designed by famed sculptor, Alexander Calder. These kinetic sculptures are challenging for users to conceptualize, as the relationships and properties that keep the pendants suspended in a particular location are not immediately apparent. Even to those who understand the relatively simple physics behind mobile balancing, it is very difficult to construct them manually, as the current state of the art involves a lot of guesswork, with plenty of room for human error. These properties, combined with the general appeal of the end product, make mobiles a perfect candidate for a computational approach. With our Unity application, users are guided through an interactive process that allows them to design their desired mobile in an intuitive, visual way. Our tool does all of the necessary background computations to ensure the mobile's desired properties, then supports the user all the way up to fabrication by outputting a laser-cuttable file that can be assembled easily and quickly using readily available materials. By making the process more intuitive and exact, our interactive fabrication tool allows users to explore the design space for 2-Dimensional Calder-style mobiles more fully.

Keywords: computational fabrication

1 Introduction

Hanging sculptures have existed within the artistic community since the 1920s when then American artist, Man Ray, created some of the first specimens. Later in the 1930s, famed sculptor Alexander Calder broke through the formerly narrow limitations of these sculptures when he introduced his first abstract works. Even at the time of their introduction, the sculptures were widely appreciated in the public sphere due to their aesthetically pleasing and uniquely dynamic nature.

Though the physical concepts behind the mobile's suspension are fairly straightforward and well understood from a technical perspective, humans are not particularly adept at predicting balancing points. This is true especially for balancing complex shapes with varying sizes, densities, etc., where even the simplest of construction problems exceeds the human capacity to intuit. As such, the current state of Calder-style mobile-making relies heavily on an iterative trial-and-error approach, by which the artist continually refines a mental estimation for a pendant's center of mass by testing the orientation of the pendant as it is supported from various points. This is often done by simply holding the pendant between the artist's thumb and pointer finger, in which case it is impossible to be sure that they are not exerting any additional force on the object which would skew its otherwise freely hanging orientation. Even once a satisfactory point is identified, the task still remains to create a connector at this point, such that the pendant will be able to suspend from it. Clearly, this process is very time- and labor-intensive, with an extensive potential for error. This renders the art of mobile making relatively inaccessible, particularly because the mobiles are generally very sensitive to inconsistencies – even a small deviation from the optimal suspension point can result in a drastic deviation from the mobile's desired pattern.

However, since the physical underpinnings of this problem are well defined, the calculations necessary to make one of these Calder designs feasible in the physical world are fairly simple to determine computationally. Coupled with the recent rise of personal fabrication technologies, a computational system that supports the user through the entire design and fabrication processes seems particularly desirable. We seized the opportunity to make this art form more accessible and customizable to the general public through visual interactive software. Our application computes the necessary physical calculations behind an intuitive user interface that allows any novice user to create physically realizable Calder-style hanging mobiles with almost no learning curve, and a minimal time investment for both the design and fabrication phases.

2 Related Work

The physical concepts underlying balancing installations are remarkably simple, but have spawned a great deal of scientific research. The research has grown tremendously over the past few years in particular, as rapid prototyping technologies have increased in both popularity and availability.

The paper from which we drew most heavily is Prévost et. al. [2013], which proposes a way to optimize 3D models via internal manipulations such that these models will balance in a particular orientation upon fabrication, while remaining as close to the user's desired visual appearance as possible. This paper primarily deals

*Liane.E.Makatura.17@Dartmouth.edu
†Catherine.Most@Dartmouth.edu
‡Gloria.Y.Li.16@dartmouth.edu

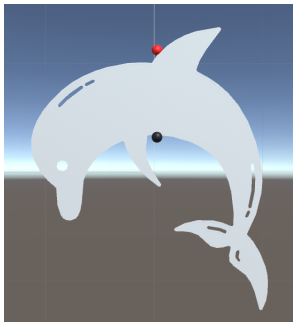


Figure 2: *Constraints for the placement of the suspension point p (red dot) for an object with center of mass c (black dot) in a specified orientation. p must be directly above c with respect to the direction of gravity, in this case $(0, -1, 0)$.*

with objects intended to stand on a flat surface, but peripherally explores the application of their method to objects intended for suspension. Our project seeks to build on the insight developed here by extending computational balance optimization to multi-body systems instead of individual 3D models.

3 Technical Approach

Due to the time constraints of our course project, we focused primarily on the creation of Calder-style mobiles with 2D shapes. That is, we focused on pendants that were fully defined by their 2-dimensional outlines, then extruded directly in the z-dimension to create a uniform material thickness. However, our calculations and construction approach are generalizable in many different ways. This section details the high-level formulation underlying our tool. Our particular implementation of these concepts is left for Section 4.

3.1 Mobile Construction

As described in Mahler's "How To" guide for mobile-making [Mahler 2014], we employ an iterative, bottom up approach to build up our structures. We begin with our base case, which seeks to balance individual pendants that are not connected to one another – that is, we aim to find a point p on the pendant mesh that allows the pendant to hang in a specified orientation when it is suspended from that point. As described in Prévost et. al. [2013], we note that this can be accomplished by ensuring that the p is directly above the pendant's center of mass c in the desired orientation with respect to the direction of gravity, as shown in figure 2.

Having satisfied this constraint, we move on to the multi-body base case, which involves the connection and subsequent balancing of 2 selected pendants. Mimicking the single-pendant base case above, we note that the suspension point for this grouping must be directly above the multi-body center of mass of the group containing the pendants and the connector which now spans them. Computing this location is straightforward as before, as we recall that an entity i can be treated as a point mass located at its center of mass, c_i. Thus, to find the center of mass for this unit, we need only compute a weighted average the three centers of mass together (where each center of mass is weighted by the mass of the object). We then shift the suspension point for this grouping vertically upward (in the direction opposite gravity) until it intersects with one of the three meshes in the group. In the case of traditional binary Calder-mobiles, this point typically falls on the connector between the two original pendants. However, there is a more general d-nary formulation of the mobile problem in which this might not be the case. Though we focus on the more common binary version in our current implementation, our approach is easily generalizable to this d-nary formulation. Each of these is discussed in greater detail below.

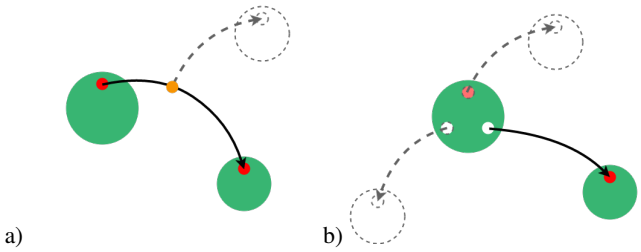


Figure 3: *Examples of (a) a suspension point and (b) a connection point, and potential outcomes. (a) Suspension points (in red) are calculated once the shape is ready to be hung from a connecting curve or another shape, and so the center of mass can be calculated for the shape. Connecting 2 suspension points creates a new suspension point (in orange) on the connecting curve itself, aligning with the new center of mass that falls somewhere in between the two figures. (b) Utilizing a connection point (in white) shifts the center of mass from the top shape to the overall unit. At this point, either the suspension point can be calculated, or another connection point can be created to balance out the mass.*

3.2 Types of Points

We have determined two different kinds of points that shapes can hang from, which allow for generalization in making many different kinds of mobiles.

Suspension Points Suspension points force shapes to follow a binary pattern, which is useful for Calder-style mobiles. These points are the specific point of suspension for a shape, which aligns with the center of mass so that the shape can hang in equilibrium. Calder mobiles derive their unique shape from exclusively using suspension points, so that every shape is suspended by a connecting wire, and not another pendant. In this approach, the center of mass of a connected unit falls in line with the topmost connector itself. The mobile hierarchy contained below this connector can be suspended from it; the mobile can now be either complete, or treated as a point mass such that it becomes a child within an even taller mobile. Once an object is connected by its suspension point, it becomes essentially immutable, since connecting another object to it by a non-suspension point would change the center of mass of the connected unit. As shown in figure 3.a, 2 shapes connected by their suspension points should only result in a new suspension point being created. Because the center of mass of the objects now lies in line with the connector, the new suspension point is created on the connector itself so that the unit can balance and hang from another object.

Connection Points Connection points follow a d-nary structure, in that multiple objects can hang from a single object. Connection points do not necessarily lie in line with the center of mass, and they usually fall below the suspension point on the object. These points allow for the creation of tiered and hanging mobiles, which might have distinct levels or perhaps a top bar that all other objects hang from. Connection points must be used before suspension points, so that the suspension point can be calculated based on the center of mass of the entire hanging unit. Top shapes can therefore have multiple connection points. As shown in figure 3.b, shapes that link

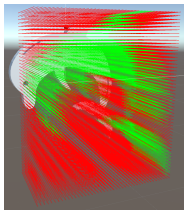
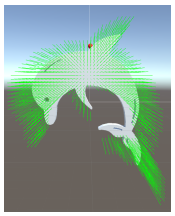
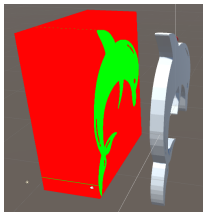
a)  b)  c) 

Figure 4: *Visualization of our sampling method for calculating center of mass, with sparse sampling (a) showing positive (green) and negative (red) hits, then isolated positive hits (b) for clarification. Image (c) visualizes the actual density at which we sampled in the app (which is 0.01 units). The density of this sampling provides great enough precision for our purposes.*

to a top shape must link by their suspension points, to ensure that the hierarchy below the top shape is balanced. Once a shape uses a connection point, 2 outcomes are possible: another connection point is created, or a suspension point is calculated.

While we have not yet directly implemented connection points in our application due to time constraints, differentiating between connection and suspension points enabled us to generalize mobile creation and differentiate Calder styles from other styles. Suspension points are more difficult to compute than connection points, so generalization and future implementation of connection points would be relatively straightforward.

4 Implementation

We opted to implement our tool using Unity 3D, because it offered a well-tested visualization and interaction platform, while also giving us a powerful scripting API and strong debugging capabilities within the Unity environment. As with any environment, there were a few limitations that had to be circumvented, such as the fact that most of the physical simulation code is approximate, and most of the interactions we were trying to utilize (i.e. object dragging, mesh import/export, and custom mesh generation at run-time) are only supported in the editor mode (when the apps are being built) as opposed to the play mode (when the app is in use). However, we were able to overcome most of these issues in our application.

Our implementation code and assets for this project can be found at `https://github.com/LianeMakatura/Balancing_Act`, which contains the meshes used, the C# scripts for physical calculation and user flow, and all assets required for the Unity framework.

4.1 Physical Calculations

As noted above, much of the physical simulation code found in Unity is approximate. This is due to the fact that it needs only be precise enough to look convincing for gaming, which is Unity's primary use case. We therefore overrode most of Unity's related functionality with our own scripts, to enable a level of precision high enough that a mobile's desired properties would be maintained once fabricated.

For this project, the main physical component to override is the calculation of the center of mass for an arbitrary mesh. Since we are working only with meshes that have uniform thickness, and assuming a uniform density in the final material, we note that the problem can be reduced to 2-dimensions (x and y). Assuming a uniform distribution of points across the object's mesh, we would be able to calculate the center of mass by simply averaging the position of all the vertices in 3D space. However, we cannot use the mesh directly for this task, because some sections of a given mesh are potentially more detailed than others, which could artificially skew the center of mass calculations due to the presence of more vertices in a particular area.

To combat this, we leverage a ray-casting approach that uniformly samples the space determined by the bounding box of the pendant. Each of these rays is cast from the camera plane toward the mesh, and subsequently labeled as either a "hit" or a "miss" with respect to the pendant mesh in question. If the ray fired from position (x,y) does hit the mesh, we add the coordinates of the ray into our running position sum $total\_pos$ (which is a vector), then we increment the number of hits, num_hits, by 1. Otherwise, we ignore the ray. After we have sampled the entire space defined by the object's bounding box, we compute the average position by calculating $total\_pos/num\_hits$. Given our assumptions, as discussed above, the resulting vector corresponds to the pendant's physical center of mass. A visualization of this process can be seen in figure 4.

After finding the center of mass, we calculate the suspension point. The suspension point must be directly above the pendant's center of mass, and it also must be located at the top-most intersection of the object along this direction, to ensure proper suspension. This is done by initializing our guess for the suspension point location as the object's center of mass: $temp = susp\_pos = c$. The z value of $temp$ is changed to be that of the camera, so that we can follow another raycasting approach to compute the suspension point. We incrementally increase the y value of $temp$ until it reaches the y value of the maximum bound in the bounding box; this moves our guess in the direction opposite gravity, denoted by $(0, -1, 0)$. At each iteration, we cast a ray from $temp$ toward the object mesh, and label it as a "hit" or a "miss" with regard to the pendant in question. If the ray is a "hit," we update our current estimate $susp\_pos = temp$, since we have found a new highest intersection point with the mesh along the desired direction. Else, we ignore the ray. By the time we have reached the upper bound of the bounding box, we know that $susp\_pos$ is correct if we proceed in this manner. The visualization from this process can be seen in figure 2.

To recursively group selected objects together in order to form the mobile, we compute the multi-body center of mass as specified in section 3.1. This group is now treated as a single multi-body pendant, which possesses its own suspension point, calculated as above. The multi-body pendant can now be connected to other parts of the mobile in the same way that a single pendant object could.

We had also hoped to incorporate an in-line physical simulation, so that users could verify their designs in equilibrium, and also experiment with how the shapes moved when they were slightly perturbed. However, we were not able to reconcile our approach within the Unity API during the allotted time, so this feature has been left as an area for future work.

4.2 Work Flow

Given Unity's limitations, some work-around is required to create physically realizable mobiles. We include prefabbed meshes in our Unity app, which require pre-processing of meshes before they can be used successfully. The Unity app allows the user to access these meshes and create a mobile that is exportable as an .obj file. Once this file is created, further post-processing is required to slice to the file into .svg format before it can be sent to the laser cutter.

Pre-processing In order to make the work flow as seamless as possible for the user, we utilize pre-selected meshes. This eliminates many possibilities for unexpected problems with user input. Our

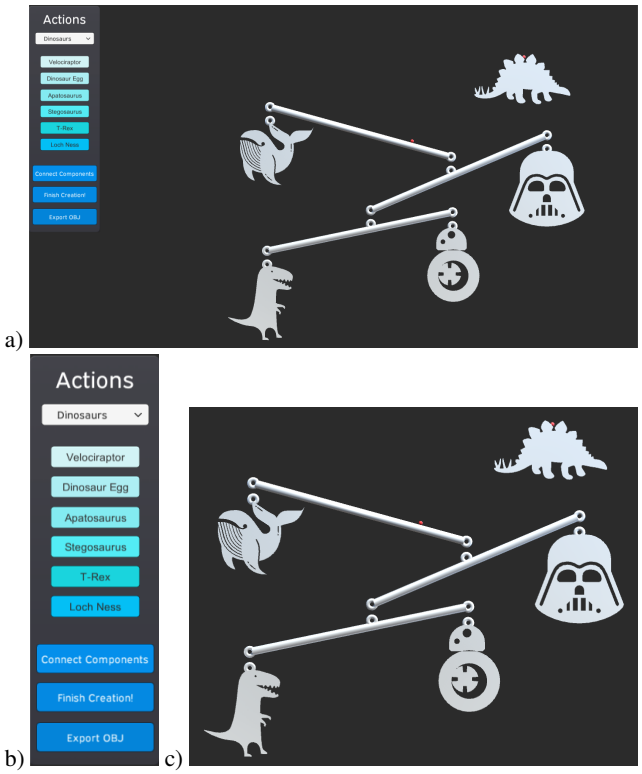


Figure 5: *(a) Screen-capture of the app in use. (b) The sidebar on the right contains a drop-down menu for shape themes, buttons for selecting different meshes, and buttons for connecting and completing the mobile. (c) Selecting a shape generates a draggable mesh with a suspension point, as seen with the Stegosaurus in the top-right corner. The user creates connecting bars and holes by selecting 2 unused suspension points (red spheres) and clicking 'Connect Components'. Currently, there are only 2 remaining suspension points: above the topmost connector, and above the newly added Stegosaurus.*

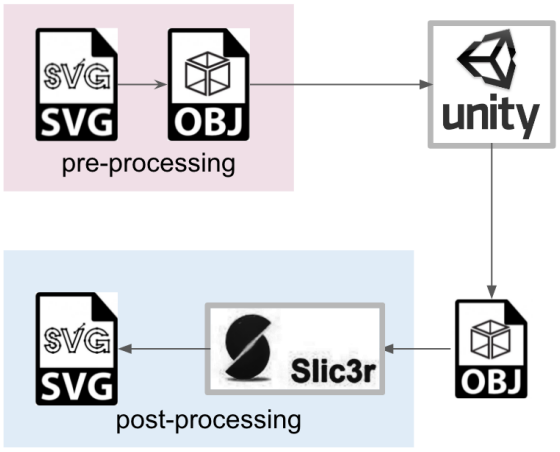


Figure 6: *The progression of files for our user flow.*

pipeline allows us to begin from a 3D model or a 2D image. The pre-processing for a 3D model begins with an .obj file of the desired pendant, which we then slice into layers along a specified direction using the freely available software, Slic3r. From the layers outputted by Slic3r, we identify the correct central layer as our desired .svg for input. Alternatively, for a 2D image we can begin directly with an .svg file of the intended shape. From the vector .svg file we input the shape into Tinkercad to easily extrude the 2D shape into a 3D shape of uniform thickness. This creates a shape with similar visual feel to the final laser cut outputs for the user and makes the final slicing step easier to compute. These .obj meshes with uniform thickness are the meshes that our app directly references when instantiating the prefabricated pendants in the GUI.

Unity app Once the .svg files have been converted into the appropriate .obj meshes, they can be input as assets into Unity. We place the meshes in a 'Resources' folder within 'Assets', which allows Unity to load them internally and display them on the screen. Each mesh is appropriately scaled so that each of them are of roughly uniform size. The center of mass and suspension points are also calculated so that a red sphere can be placed at the appropriate suspension point, which is just above the mesh and in line with the center of mass.

The user begins using the application and accessing the different meshes through the sidebar. The sidebar within the application contains a drop-down menu, 6 buttons corresponding to different meshes, and buttons for connecting components, completing mobile creation, and exporting a .obj of the mobile. All buttons are muted on start, but become un-muted by choosing a theme from the drop-down menu. The three themes - marine animals, dinosaurs, and Star Wars - contain 6 meshes each, and clicking a theme populates the 6 buttons below with the mesh options pertaining to that theme. Clicking a mesh button generates the mesh and its suspension point in the middle of the screen, which the user can drag into the desired position. The suspension point is defined relative to the pendant frame, so it moves appropriately as the mesh is dragged through the space. To create connecting bars and toruses that will allow for construction after fabrication, the user selects 2 suspension points from 2 different meshes and clicks 'Connect Components'. The resulting bar contains its own suspension point, which is aligned with the center of mass of all previous connections. Once the meshes are connected, that section and all sections below it in the hierarchy become immovable.

Building mobiles with a bottom-up approach ensures that at any point, the mobile can be hung from the top bar's suspension point. Whenever the user decides that their mobile is complete, they can click 'Finish Creation' to automatically generate a final torus around the suspension point of the top bar. At this point, all mesh buttons and the drop-down menu become muted. The mobile can now be exported with all shapes, connectors, and holes in place. Clicking 'Export OBJ' brings up a wizard allowing the user to select options for exporting. Exporting functionality is provided through the 'OBJExporter' package from the Unity Asset Store, which has been modified to be accessible by the user at run-time, instead of in its default location in developer's Unity editor.

Post-processing: The mobile designed by the user in Unity is output as a 3D .obj file centered around the xy-plane (at depth $z = 0$) for easy slicing of the 3D object into a 2D .svg file. The slicing algorithm (provided by Slic3r) outputs the multiple slices of the .obj file in one .svg file that needs to be cleaned up in a product such as Adobe Photoshop or Illustrator, so that it only includes the single central slice which will have all of the fittings for the connectors and pendants. With a single .svg file output containing the correct 2D slice of the hanging mobile, the user is nearly ready to submit their project to a laser cutter for fabrication. Depending on the

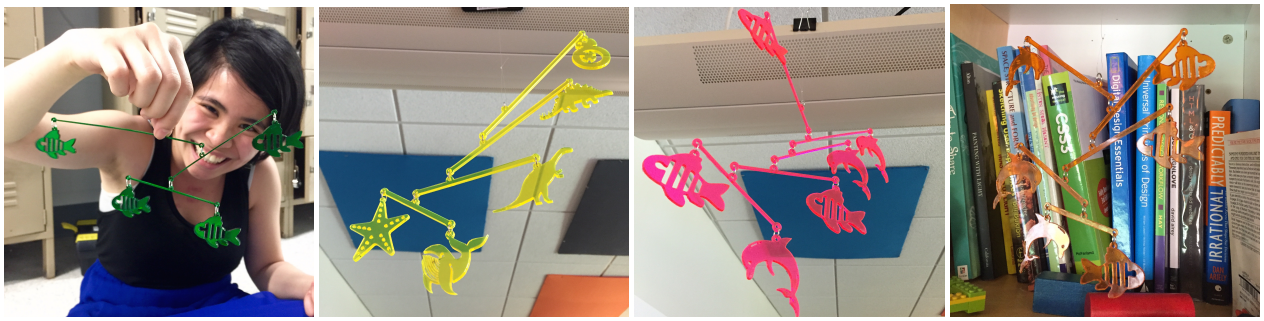


Figure 7: *Successful user fabricated hanging mobile results*

type of laser cutter, the .svg file might need to be converted to the universal file type of .dxf; this is required particularly in the event that the user must work with older or more file-limited laser cutting software.

We acknowledge that this file-transfer pipeline is somewhat convoluted, and we would like to simplify this process as part of our future work to make the application more directly usable.

4.3 Fabrication

The fabrication process begins with simple file input, of the .svg obtained in the previous steps. Our application was designed for 1/8-inch thick acrylic sheets, due to the fact that plastic has uniform thickness (as opposed to wood). It also allowed users to have the most flexibility in the visual customization, as it is readily available in a wide array of colors and other special properties, like mirrored backings or a glittery appearance. The application output pattern exactly mimics the Unity visualization, so after the pieces have been punched out of the sheet, the remnants are able to serve as an assembly guide. This helps the user as they are connecting each laser cut part to one another with metal jump rings.

Over the course of multiple user tests we were able devise a power of 100 and a speed of 1.2 as the best settings to use with a LaserPro C180 Desktop laser machine (which is what was easily available for our testing). Experimentation with individual machines may be required to ensure that the pieces are properly, safely, and fully severed.

5 Results

By thoroughly prototyping in the beginning stages of our project, we were able to determine slight differences in the angles, due to the fact that we didn't account for mass introduced by connection toruses or jump rings. Also, we were able to make adjustments to the support bars to be thicker in order to support the full weight of the pieces which we were not accounting for. By the end of the project, these iterative improvements allowed us to create mobiles that matched the design specs in Unity almost identically. Some of our final creations can be seen in figure 7.

User tests also provided insight into the intuitive nature of the application. An experienced user first designed their own mobile as the user tester watched, and then the user tester was allowed to go through the application and design a mobile. This method showed promising results, with all users excited to design and later see their own mobile. The main drawback of mobile creation was the lack of knowledge about Calder mobiles, which led to some initial confusion about what kinds of mobiles could be created - however, most users were able to quickly gain an understanding of what mobiles

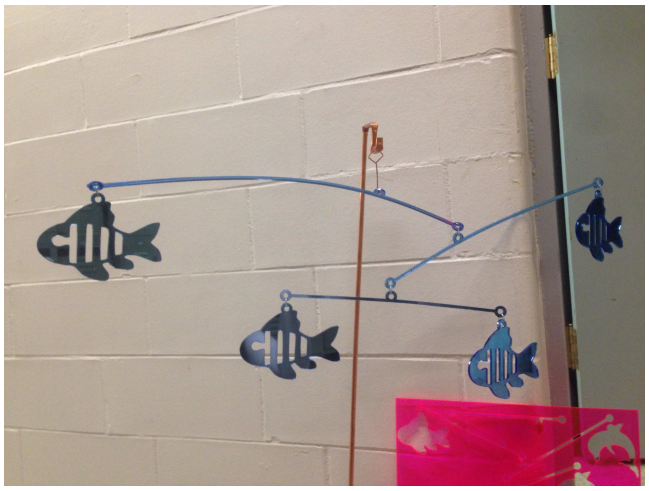


Figure 8: *Larger objects caused the original connectors to sag, seen especially in the top connector.*

they could create, and the best ways in which to create them.

In addition to our tool's confirmed intuitive nature, the physical results from our formative user studies and demos were very promising. All were user designed mobiles which came out as fully fabricable balanced hanging mobiles without any structural problems.

6 Limitations and Future Work

Our application accomplished our primary goal, which was to create software that would successfully generate balanced mobiles based on user-input shapes and placement. Limitations within our application mostly stem from the lack of time to implement, as well as Unity's inherent limitations. Users can create mobiles only from prefabbed meshes, and the mobiles that the app can create are primarily Calder-based. Unity does not provide support for easy input of SVGs or other files, and assets that allow for file input work only from the Unity editor. In-app file inputting is therefore not implemented. Moreover, file outputting was only possible through 3rd-party assets, and only exportable as OBJ. As laser cutters do not accept OBJ files, actual mobile cutting requires the user to slice SVGs from the OBJ file, adding another level of inconvenience.

User flow has its own limitations, partly due to time constraints and partly due to Unity's lack of support for direct mesh manipulation. The user cannot resize or delete meshes once the meshes are selected, and if shapes and connectors directly intersect they will not be separated in the post-processing slicing. Unfortunately

this means that meshes must be placed, so that connectors do not intersect the meshes or each other.

One limitation that came about through fabrication testing was the weight limit of the connectors themselves. Smaller mobiles created very fragile connectors which broke easily, and as shown in figure 8, larger mobiles created heavier shapes that caused longer connectors to sag heavily. We solved this problem by increasing the thickness of the connectors from 0.1 to 0.2.

Future work would focus on addressing the limitations of file input and output within the app. Implementation of connection points would also be possible, since suspension points are calculated and click-able. User control over connection points would be the main focus of this implementation. In the same vein of generalizing mobile creation, perhaps the most difficult work we would like to undertake in the future would be creation of curved connectors, either as prefabbed meshes or as user-created Bezier splines. Much of Unity's limitations revolve around lack of support for direct user manipulation, which would hold true for spline creation.

7 Conclusion

This project explored the design and fabrication space of balancing Calder style mobiles. The challenges we set out to tackle were very user centric with the goal of seamless file conversion and user control of shapes, while also featuring a computational component that had to respect physical constraints of the desired mobiles. In the short four weeks we had to accomplish these challenges we were able to make great steps towards the goal of seamless integration. Users were delighted by the interface and the simple design of the interface. There were more requests for fully fabricated mobiles from users than we were able to fabricate. Users understood and appreciated the novelty and balancing magic the application was able to do for them on the fly. The simple laser cut-table vector file output was easily understood and intuitive to the users for easy assembly.

Acknowledgements

We thank Professor Emily Whiting and our peers in CS89/189: Computational Fabrication for their guidance and helpful comments. Our peers in the DALI Lab also showed great interest in the project throughout the term and gave valuable feedback as user testers. We thank Tim Tregubov as well for providing valuable assistance with project conceptualization and technical Unity support.

References

MAHLER, M., 2014. From the artist: How to make a real mobile. http://www.houzz.com/ideabooks/34452702/list/from-the-artist-how-to-make-a-real-mobile. Accessed: 2016-06-2.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make It Stand: Balancing shapes for 3D fabrication. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH) 32*, 4, 81:1–81:10.