PhysicsBook: A Sketch-based Interface for Animating Physics Diagrams

Salman Cheema
University of Central Florida
Orlando, FL
salmanc@cs.ucf.edu

Joseph J. LaViola Jr.
University of Central Florida
Orlando, FL
jjl@eecs.ucf.edu

ABSTRACT

We present PhysicsBook, a prototype system that enables users to solve physics problems using a sketch-based interface and then animates any diagram used in solving the problem to show that the solution is correct. PhysicsBook recognizes the diagrams in the solution and infers relationships among diagram components through the recognition of mathematics and annotations such as arrows and dotted lines. For animation, PhysicsBook uses a customized physics engine that provides entry points for hand-written mathematics and diagrams. We discuss the design of PhysicsBook, including details of algorithms for sketch recognition, inference of user intent and creation of animations based on the mathematics written by a user. Specifically, we describe how the physics engine uses domain knowledge to perform data transformations in instances where it cannot use a given equation directly. This enables PhysicsBook to deal with domains of problems that are not directly related to classical mechanics. We provide examples of scenarios of how PhysicsBook could be used as part of an intelligent tutoring system and discuss the strengths and weaknesses of our current prototype. Lastly, we present the findings of a preliminary usability study with five participants.

Author Keywords

Sketch-based User Interfaces, Sketch Recognition, Mathematical Sketching, Inferring User Intent

ACM Classification Keywords

H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General Terms

Algorithms, Human Factors

INTRODUCTION

Sketch-based interfaces capture the ease of using pen and paper to communicate ideas while simultaneously leveraging the power of computation. Such interfaces have been used for tasks such as 3D modeling [8], musical score creation [6], and website design [16]. We are interested in the application of sketch-based interaction in physics and mathematics domains. Both teachers and students often sketch diagrams while solving problems or demonstrating concepts. When students are asked to solve physics/mathematics problems, their solutions usually follow a general pattern. They start from a problem statement, and usually begin by drawing a diagram and annotating it with the information provided in the problem statement. Common annotations for physics diagrams include shape labels, arrows indicating vector quantities and the scale of motion. Students then go through a series of mathematical steps to arrive at an answer, typically a function or number. On paper, the answer itself does little to impart intuitive knowledge to a student. Students have to rely on their imagination to visualize how their answer would affect the scenario presented in the question. Figure 1 illustrates such a scenario.

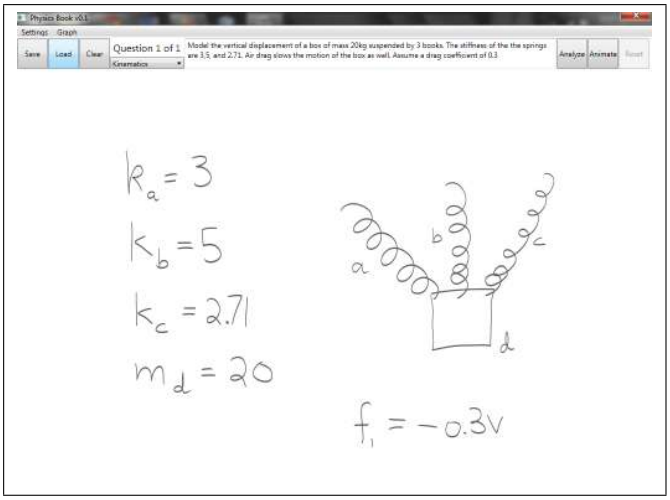


Figure 1. An example scenario where a student wants to model the vertical displacement of a box suspended by three springs and affected by air drag. Using PhysicsBook, the student can view the vertical displacement by sketching the system as shown and viewing the graph during animation.

A sketch-based tutoring system promises a natural mode of interaction, while leveraging the power of computation to animate the sketch. To this end, it is vital to develop an understanding of the entire problem-solving process, including the statement of the question, the mathematical steps in the solution and any sketched diagrams along with their annotations.

IUI'12, February 14–17, 2012, Lisbon, Portugal.

Our research goal is to construct a sketch-based physics tutoring system that integrates such knowledge and uses it to facilitate student learning. We envision it to have the following feature set:

1. *Natural Interaction*: Allow students to solve a problem as they would using pen and paper.
2. *Robust Recognition*: Distinguish between mathematical steps in a given problem's solution and any diagrams that students may have sketched.
3. *Reasoning*: Should be capable of high-level reasoning about diagrams and the mathematical steps in a student's solution . This can be done by integrating a sufficient level of physics knowledge into the system. Such reasoning allows inference of intent behind a sketch and can be used to provide error feedback and hints to students.
4. *Feedback*: Allow students to view how the values of physical quantities vary during animation. Graphing is an important tool in this respect. Step by step feedback and error highlighting may also be used for knowledge scaffolding [24].
5. *Seamless Animation*: The tutoring system should be able to animate the diagrams in a student's solution, given any granularity of input. Triggering the animation should be seamless and natural. Students should be able to interact with and manipulate the animation of sketched diagrams, by use of hand-written mathematics and/or gestures.

With these goals in mind, we have developed PhysicsBook, a prototype sketch-based system that lets users write down the solution to a physics problem in a natural manner, including mathematics and free-form drawings. Using the answer to a problem, PhysicsBook animates the drawing to indicate whether the solution was correct. Existing interaction techniques such as mathematical sketching and the MathPad^2 system [13], were used as a starting point in the design of PhysicsBook. However, PhysicsBook goes well beyond these ideas because it does not require mathematical definitions of how objects should animate as functions of time. Instead, by integrating knowledge of classical mechanics and its related concepts, PhysicsBook is able to construct animations for a range of physics problems, with just the sketched diagram and a minimum amount of mathematics needed to solve the problem.

RELATED WORK

Sketch Recognition

Understanding of free-form sketches is a hard problem because of variation in symbols and notation in different domains. Paulson et al. [20] describe techniques for recognizing and beautifying several low-level sketch primitives. Patel et al. [19] have conducted a statistical analysis to select the most appropriate ink features for distinguishing between shapes and text in sketches. Hammond and Davis have developed LADDER [7] which allows a user to specify sketch recognition rules in a domain independent manner. Users can describe sketch primitives and some high level semantics for individual domains in text form, which is then used to generate domain specific recognizers. User perception of sketch recognition has been explored by Wais, Wolin, and Alvarado [23], who draw some very important lessons for sketch-based systems in general.

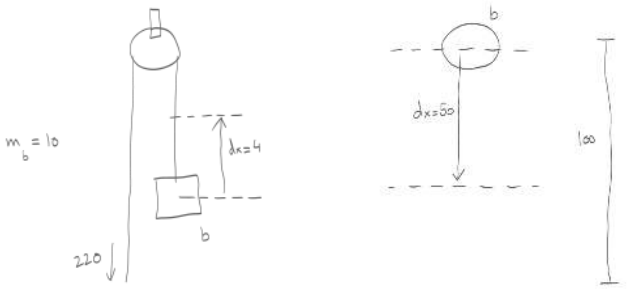
Higher level sketch-based applications have also been developed for a variety of contexts. MathPad^2 [13] provides a method for integrating written mathematics with animation of hand-drawn sketches. It is a domain independent system, where users must specify all aspects of animation through mathematics. PenProof [9] is a pen-based geometry theorem proving system that can infer correspondences between geometric constructs and proof steps. CogSketch [5] is an open domain sketch analysis system, which enables conceptual labeling of sketches. ChemInk [18] is a sketch recognition system for chemical drawings that uses conditional random fields to deliver good recognition accuracy. None of these applications have intelligent tutoring in physics as their motivation.

Physics Tutoring Systems

Traditionally, physics tutoring systems have relied on WIMP interfaces. A representative example is the Andes Physics Tutoring System [22], which provides step by step guidance in problem solving. The use of WIMP interfaces has several drawbacks. Students cannot write down their solutions in a natural manner as if using pen and paper. Tutoring systems in general must utilize some understanding of a given problem and its solution, in order to provide feedback. An advantage of sketch-based tutoring systems is the promise of a more natural mode of interaction. Compared to WIMP based interfaces, they can also incorporate sketch understanding techniques to provide better feedback/animation mechanisms.

While no sketch-based tutoring system of sufficient capability exists for physics, several researchers have made strides toward one in recent years. Alvarado [2], Oltmans [17] and Kara [11] have demonstrated systems for sketch understanding in the domains of computer aided design, mechanical design and vibratory systems. These tools can recognize and animate relevant diagrams but none of these allow users to write down mathematics that can influence animation. Newton's Pen [14] is a simple pen-based tutoring system for statics that does not allow free-form drawing or the use of written mathematics in animation.

Steps toward integration of mathematics and sketched physics diagrams, in order to generate animation have been taken by Cheema and LaViola [3, 4]. In particular, [3] investigates the use of Mathematical Sketching [10, 12] as an interaction mechanism for physics tutoring, enabling animation of simple diagrams with different granularities of input. This system employs sketch correction mechanisms for ensuring correct animation. However, it is limited to basic concepts related to motion such as the use of position, velocity, acceleration and force variables defined as functions of time. Additionally, it does not support implicit association of written mathematics nor allows the use of diagram annotations to infer the user's intent. The use of such annotations is extremely important to problem solving in physics [21, 27]. One major advantage of PhysicsBook over [3] is the use of data transformations that allow it to deal with instances where the given problem solution cannot be directly used for animation.



(a) Work done by pulley (b) Change in potential energy during freefall

Figure 2. A set of sample problems selected from a physics textbook.

Lastly, several commercial tools are also available that let users construct animations for physics concepts. Representative examples include Algodoo [1] and Working Model 2D [25]. Such tools can create animations but they do not let students work out a given problem and then directly associate the answer with a diagram to do the animation. In such tools, a student would solve the problem in a notebook and then have to separately reconstruct the diagram using the tool to perform the animation.

CHALLENGES IN UNDERSTANDING USER INTENT

The goals outlined in the introduction imply the need for core components to do three main tasks: Recognition, Inference and Animation. For sketch understanding, the most important tasks are Recognition and Inference. A recognizer accepts a sketch as input, containing both diagram and mathematics and generates a collection of primitive components which are then analyzed to infer correlations between the recognized diagram, its annotations and the mathematical equations in the solution. An important aspect of this process is the inference of intent in ambiguous cases. This encompasses understanding and beautification of approximate sketches and the use of annotations to gain a deeper understanding of the given problem. Techniques for beautification of sketches have been described for low-level primitives by [20] and for limited cases of higher level primitives by [3].

The challenges in sketch understanding can be highlighted by examining a set of questions from a physics textbook. In figure 2(a), a box of mass 10kg is lifted to a height of 4m by applying a force of 220N over an ideal pulley. The student is asked to find the work done on the box. A possible diagram in this case includes a circle denoting the pulley, with straight lines indicating wires. An arrow labeled with a numeric quantity can be used to indicate the force and an arrow sketched between two dotted lines can indicate change in displacement. It should be noted that this representation is not unique. The wire over the pulley could have been sketched as a single line passing over the circle, instead of as two separate line segments. Also, the student has indicated that the circle at the top is to serve as the pulley, by drawing an overlapping square to act as the hinge.

Similarly, Figure 2(b) depicts a scenario where a ball of mass 15kg is dropped from a 100m height. The student is asked to find the change in gravitational potential energy of the ball after it has fallen 50m. Note that the distance scale drawn to the right side of the sketch is redundant. The sketch contains sufficient information for animation without it. Yet, students commonly associate redundant information with a sketched diagram, in order to keep track of given information. The use of dotted lines and an arrow to indicate different levels of displacement here is common with Figure 2(a). Both examples highlight the importance of developing a semantic understanding of annotations for the animation of different classes of physics diagrams. It should also be noted that in both problems, some information is missing (e.g., the student did not indicate weight).

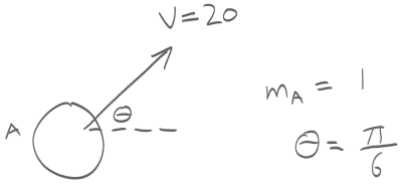


Figure 3. A soccer ball of mass 1kg is kicked with a velocity of 20m/s at an angle of $\frac{\pi}{6}$ with the horizontal. Find maximum height achieved by the ball.

Figure 3 presents a simple projectile problem that asks for the maximum height of a kicked soccer ball. A student sketches a circle to indicate the ball, and annotates it with an arrow to indicate initial velocity. The arrow is further supplemented by a dotted line that indicates the initial angle of the trajectory with the horizontal axis. Equations indicating values for mass and horizontal angle are also included. The representation contains enough information for an animation, but at the same time, there are no annotations indicating the highest point of the trajectory. This could be added in the form of a dotted line or an arrow to indicate the maximum displacement in vertical direction. This example also highlights the use of two annotations (arrow and dotted line) to define a vector quantity.

These examples highlight some of the challenges of sketch understanding for the domain of classical mechanics problems. There are often multiple ways of drawing a diagram to accomplish the same task. Students can annotate and label the diagram with arrows, dotted lines, symbols, numbers and even equations. Similar annotations carry different meaning in relation to different sketched shapes, and in relation to problem type. The information contained in sketched diagrams may be incomplete or even redundant in some cases. Such scenarios can be confusing from a recognition standpoint. Lastly, user sketches may be approximate in nature. For example, consider the problem shown in Figure 2(a). The wire to the right side of the pulley does not actually touch the square. An incorrect animation will result if the inference subsystem cannot link it to the shape properly.

A recognition system for a sketch-based physics tutor must be able to meet these challenges, in order to generate a good understanding of the problem. An understanding of a student's written solution (containing mathematics and diagram) can be

used to provide hints and highlight errors, while facilitating animation. Furthermore, the use of sketch-based interfaces should allow the students to integrate the answers to assigned problems into the animation process, in order to see if a given answer causes an intuitively correct animation.

PHYSICSBOOK INTERACTION MODEL

In PhysicsBook, users sketch diagrams and write mathematics with a stylus on a tablet computer. Users can make associations between mathematical expressions and recognized sketch primitives by selecting mathematical expression(s) with a 'Lasso' gesture and then tapping a recognized item. The recognition system does real-time recognition of mathematics and can provide feedback for recognized mathematics. However, viewing of real-time feedback is disabled by default[1]. Users can trigger sketch recognition after they finish solving the problem. This approach is taken due to the finding by [23], who discovered that users prefer to trigger sketch recognition after they are done with drawing. Math recognition is conducted in real-time because of its usefulness in implicit associations processing. The system can recognize the following diagram primitives: convex shapes (circles, polygons), springs, wires and simple pulleys. After recognizing primitive components, the system attempts recognition of diagram annotations: arrows, distance scales, horizontal and vertical axes (dotted lines), and displacement level indicators (dotted lines with an arrow indicating a transition). Each primitive component's attributes (e.g. mass, spring stiffness, etc) are assigned initial values based on its appearance and annotations. This allows user input to be flexible because the system does not require the full mathematical description of all initial conditions and can provide a good animation given any granularity of input.

For each annotation, the system tries to infer its label by means of a proximity check. Labels can be symbolic, numeric or equations. Numeric labels are assumed to denote the magnitude of a physical quantity. The related variable in such cases is established by inferring user intent behind the annotation. For example, in Figure 2(a), the arrow indicating the force acting on the pulley has a numeric label. Taken in isolation, it indicates a vector quantity with a magnitude of 220. Taken in context of the pulley, PhysicsBook can reason that the arrow denotes a force acting on the free end. On the other hand, if an annotation's label is symbolic, the system attempts to infer the related equation by finding a match within the mathematics written down by the user. If an equation is associated with a label, it is evaluated to yield the variable in question. Users can alter initial values by writing down mathematical expressions to reflect proper initial conditions and associating them with diagram primitives. A 'Lasso' gesture is used to select mathematical expression(s). The association is completed by tapping the desired component. Existing associations can be viewed by hovering the stylus over a component.

Mathematical expressions can either be constant expressions (e.g. $m_A = 25$) or equations (e.g. $v_x = k\sin\theta$). Vector quantities can be specified either in terms of x and y component equations or as a single vector equation. If x and y components are not specified in an equation, we use physics domain knowledge to determine if the variable on the left-hand side of the equation is a vector or a scalar. For example, if a user writes $v_2 = v_1 + at$, the system can infer that since the variable v is often used to indicate the velocity of a moving body, the user has written down an expression for velocity. In cases, where the variable itself does not provide a hint, the parameters of the equation are examined to infer whether the quantity is vector or scalar. For example, if a user writes down $z = v_1 + gt$, PhysicsBook will be able to infer that v_1 has been used as a parameter instead of $\|v_1\|$, therefore z should be a vector quantity. PhysicsBook can deal with equations including both numeric (e.g. $f_1 = -0.5v_A$) and symbolic parameters (e.g. $f_1 = -\mu v$ where $\mu = -0.5$). By integrating physics domain knowledge, the system also avoids erroneous associations. An example of an erroneous association is trying to associate an equation specifying momentum with a spring. Illogical cases such as these are ignored by PhysicsBook.

Recognition errors or actual mistakes can cause incorrect animation that conflicts with a user's intuition. A 'Reset' mode is provided that lets users debug and correct existing equations and associations. This allows users to experiment with different initial values and gain better insight into the working of underlying concepts. Users can also view a real-time graph of several attributes of a given component. In keeping with our goal of natural interaction, the system preserves existing associations along different system modes. If users wish to alter part of an equation that is already associated with a component, they do not have to make the association again.

SYSTEM DESIGN

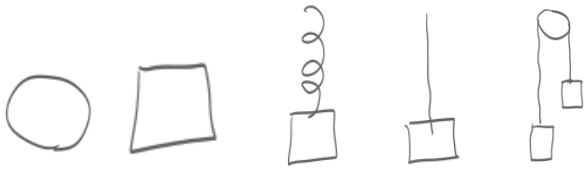
PhysicsBook has three main subsystems: Recognition, Inference and Animation. A sketch containing mathematics, diagram(s), and annotations forms the input to PhysicsBook. This sketch is recognized to yield diagram primitives, which are then analyzed and correlated with their annotations. Users can manually make associations between recognized diagram primitives and mathematical expressions. The end result is an animation. The animation subsystem also includes functionality for viewing a realtime graph of one or more interesting attributes of individual diagram components.

Sketch Recognition And Inference

A sketch is acquired as a collection of digital ink strokes, each of which is a sequence of 2D points. We use the ISTraw algorithm [26] to enumerate all cusps in each ink stroke. For recognition of mathematical expressions, we use StarPad, which is based on [28]. StarPad can recognize mathematical expressions and provide feedback in real-time. Recognition results are displayed next to hand-written expressions. When a user triggers sketch recognition, primitives such as shapes, wires, springs and pulleys are recognized first. Next, annotations such as arrows, distance scales and displacement indicators are detected. The third step consists of finding labels for recognized primitives and annotations. The final step links related labels and equations with their corresponding sketched

[1]Users can enable it via a settings window

primitive. Figure 4 shows examples of the sketch primitives that can be recognized by PhysicsBook, while Figure 5 shows the annotations that can be applied to recognized sketch primitives.



(a) Circles (b) Polygons (c) Springs (d) Wires (e) Pulleys

Figure 4. Examples of diagram components that can be recognized by PhysicsBook.

For recognizing circles, polygons, wires and springs, we based our algorithms on [3]. However, we modified the circle recognition algorithm to use the variance in average radius about the centroid of an ink stroke, instead of the variance in average angle subtended at the centroid (as suggested by [3]). The advantage to this approach is to have a more human readable threshold value supporting a much higher recognition accuracy. Any shapes that overlap each other are set aside as composite primitives. After the recognition of simple primitives, the system tries to recognize composite primitives. Currently, pulleys are the only supported composite primitive in PhysicsBook. They consist of a circle overlapped with a rectangle. The rectangle indicates the hinge of the pulley and serves to disambiguate pulleys from being recognized as other shapes.

(a) Arrows (b) Axes (c) Displacement Indicators

Figure 5. Examples of annotations in PhysicsBook that can be applied to recognized diagram components.

Implicit Associations

The function of the inference subsystem is to take recognized diagram primitives and annotations, to infer associations between them and establish initial values for variables which are then used for animation. Algorithm 1 shows the workflow for this module. First, labels and related equations are inferred for recognized annotations. Next, the relationships between annotations and sketch primitives are inferred implicitly, based on their spatial proximity. After this, labels and equations for sketched primitives are found. Lastly, equations corresponding to associated labels are found and associated with appropriate sketch primitives implicitly. This eliminates the need for users to associate all required initial conditions with the diagram manually. Implicit mathematical associations, formed in this manner, are then analyzed to extract information related to the initial state of sketch primitives.

Algorithm 1 Pseudo code for Recognition and Inference subsystems

Require: InkStrokes
  $Primitives \leftarrow RecognizePrimitives(InkStokes)$
  $Annotations \leftarrow RecognizeAnnotations(InkStokes)$
  for all Annotation $A_i \in Annotations$ **do**
    Try to find and associate label
    Find and associate related equation
    Find primitive with which this annotation is associated
  end for
  for all Primitive $P_i \in Primitives$ **do**
    Try to find and associate label
    Find and associate related equations from written equations
    Associate related equations from annotations
    Assign initial values to attributes
  end for

Data Extraction from Implicit Associations

After an association has been made (either implicitly by the inference subsystem or explicitly by the user), the system has to infer the actual value of the quantity denoted by the association. Annotations denoted by arrows indicate vector quantities. Arrows can have dotted lines associated with them to indicate the angle with vertical or horizontal axes. By examining arrow labels, in combination with associated angle and axis annotation, the system is able to infer any of the following vector quantities: velocity, force, momentum, acceleration.

Displacement indicators (an arrow drawn from one dotted line to another dotted line) are useful to indicate the direction of motion. They also indicate the starting and ending levels for the motion of a moving body. If an equation specifying the exact amount of displacement is specified, they also serve as distance indicators (useful to map distance in pixels to real world units). During the data extraction phase of inference, displacement indicators for each sketched shape are found (if they are part of the sketch) and associated. These are fundamentally important from an animation perspective. They indicate the correct starting and ending levels of motion from a user's perspective. Therefore, if the associated answer is incorrect, the shape will not move the requisite distance, making it obvious to the user that there is something wrong with his/her solution. In other words, by drawing displacement indicators, users explicitly indicate the limits of expected motion, which is extremely useful as the system does not have to guess them.

Animation

PhysicsBook uses a customized 2D physics engine based on concepts described in [15]. The physics engine includes standard functionality such as collision processing, application and resolution of different types of forces and an incremental position update mechanism. However, unlike a typical physics engine used in video games, it supports recognized sketch primitives as inputs. Additionally, it provides entry points for hand-written mathematics that can be associated with the physical properties of components in the physics engine, altering their animation behavior. Any such mathemat-

Algorithm 2 Position update mechanism for each shape

Require: Shapes, Associations
  for all Shape$S_i \in Shapes$ **do**
    if Equation $E_{position}(S_i) \in Associations$ **then**
      $Position(S_i) \leftarrow Evaluate(E_{position}(S_i))$
    else if Equation $E_{velocity}(S_i) \in Associations$ **then**
      $Velocity(S_i) \leftarrow Evaluate(E_{velocity}(S_i))$
      $Position(S_i) \leftarrow Integrate(Velocity(S_i))$
    else
      Compute $\Sigma Forces(S_i)$
      $Acceleration(S_i) \leftarrow \frac{\Sigma Forces(S_i)}{Mass(S_i)}$
      $Velocity(S_i) \leftarrow Integrate(Acceleration(S_i))$
      $Position(S_i) \leftarrow Integrate(Velocity(S_i))$
    end if
  end for

ical equations are evaluated at runtime, as needed, to yield scalar or vector values which are used to generate an animation that is concordant with a user's intent. It should be noted that the modified behavior caused by associated mathematics may or may not be intuitively correct. This serves as a feedback mechanism to the user indicating whether there is a mistake in his/her solution.

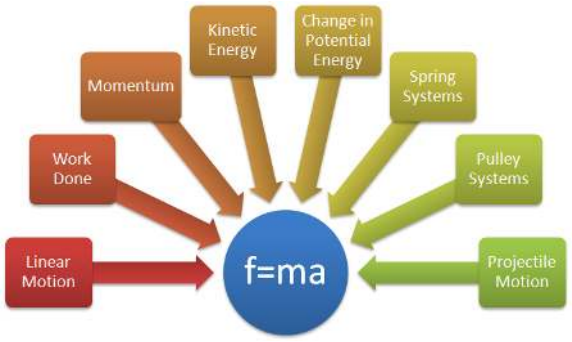


Figure 6. Figure showing physics concepts that are related to $f = ma$. Some of these concepts can be used directly as input to the animation system. For others, we incorporate data transformations that are invisible to a system user, giving much more functionality.

Algorithm 2 shows the steps taken to update the position of a shape in each animation frame. It shows that users can specify their own equations for forces, velocity, position, and acceleration directly which can augment or even replace the standard update mechanism of the physics engine. This scheme is highlighted in Algorithm 2 which shows that if a user directly specifies the position update equation for a shape, then the system's regular flow is bypassed entirely. Alternatively, if a velocity equation is specified, the system does a single integration step to update the position. If position or velocity equations are not specified, then the net force on each shape is computed, followed by the acceleration. Two integration steps are then required to update the position. Individual forces acting on a body may also be governed by associated mathematical equations, which can affect the Compute $\Sigma Forces(S_i)$ step in Algorithm 2. The advantage to using an open-ended approach like this is that it keeps the design of the animation system flexible, while allowing users freedom to associate their own equations with several physical aspects of components, which in turn enables the animation of a variety of physics problems.

Variable	Transformed To
Force	Self
Acceleration	Self
Velocity	Self
Position	Self
Spring Stiffness	Self
Wire Tension	Self
Momentum	Velocity
Work Done	Displacement
Change in Potential Energy	Displacement
Kinetic Energy	Velocity

Table 1. Variable that can be associated with diagram components in PhysicsBook. The table shows the transformation performed for each variable.

Variable Transformations

PhysicsBook allows users to animate problems from a range of physics domains related to classical mechanics. These are all concepts related to $f = ma$ (See Figure 6). However, the physics engine used for animation only accepts associations that affect simple attributes of diagram components (e.g., position, displacement, velocity, force, acceleration, and mass). Concepts that are indirectly related to $f = ma$ (e.g., work done, kinetic energy, and potential energy) must be transformed into one of the acceptable inputs, before animation can proceed. This is a deliberate design choice to keep the animation system simple, while enabling the animation of a large range of problems. Table 1 shows the list of variables which can alter animation behavior in PhysicsBook. It also highlights the transformed variable if the original variable cannot be used directly. For example, kinetic energy cannot directly affect a shape's attributes in the physics engine. However, using its formulation, the magnitude of velocity from a shape's kinetic energy can be derived as

$$K_e = \frac{1}{2}mv^2 \Longrightarrow \|v\| = \sqrt{\frac{2K_e}{m}}.$$

Coupled with an arrow annotation that indicates the direction of velocity, we can map kinetic energy to the velocity of a shape. Similarly, an annotation indicating the direction of motion (can either be an arrow or a displacement indicator) can be used in conjunction with an expression denoting the work done or change in gravitational potential energy for animation. In either of these cases, the system will infer the magnitude of required displacement for the annotated diagram component using one of the following equations:

$$W = F_1.\Delta x \Longrightarrow \|\Delta x\| = \frac{W}{\|F_1\|}$$

$$P_e = mg.\Delta h \Longrightarrow \|\Delta h\| = \frac{P_e}{m\|g\|}.$$

This mechanism for transforming variables is useful because it allows us to extend our animation subsystem to concepts that are related to $f = ma$ (See Figure 6) but not covered

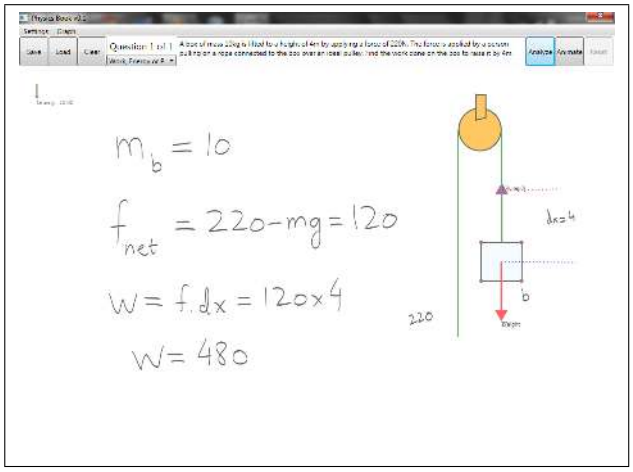


Figure 7. Example scenario where a box of mass 10kg is lifted to a height of 4m by applying a force of 220N. The force is applied by a pulling on a rope connected to the box over an ideal pulley. The student is asked to find the work done on the box as it is raised by 4m.

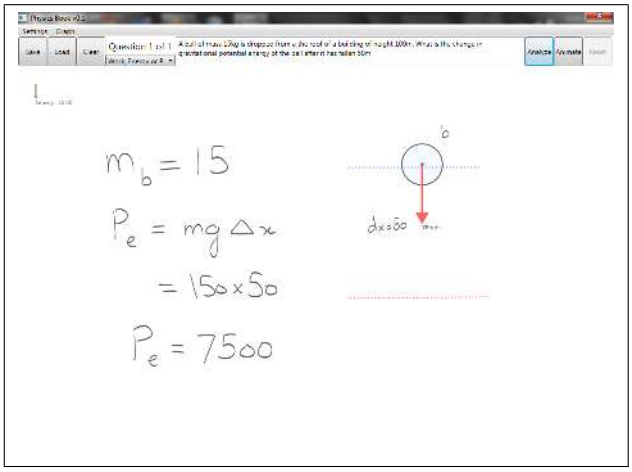


Figure 8. Example scenario where a ball of mass 15kg is dropped from a the roof of a building of height 100m. The student is asked to work out the change in gravitational potential energy of the ball after it has fallen 50m.

by any of the allowed inputs to our physics engine. It should be noted that in order to fully support related concepts, data transformations alone are not sufficient. The transformations described here highlight the fact thats diagram annotations play a key role in enabling animation for each of these instances. In all three instances described, transforming the variable yields only the magnitude of the transformed quantity. In order to get the direction for each of them, the system needs an arrow annotation to be drawn by the user.

This also illustrates the importance of diagram annotations for a sketch-based intelligent tutoring system. Diagram annotations are a natural interaction mechanism that is frequently used by students when working with pen-and-paper. Annotations allow our system to implicitly associate initial values based on their proximity to sketched primitives, freeing the user from having to make the association explicitly. Lastly, annotations aid our system in variable transformations, allowing it to guess the correct direction for vector quantities that have to be transformed from concepts that are not directly related to $f = ma$.

EXAMPLE SCENARIOS

Work Done by Force Acting through a Simple Pulley

Figure 7 represents a scenario that involves a box being pulled upward by means of a pulley. The information given to a student is the mass of the box and the magnitude of the force acting on the pulley. The student is asked to find the work done on the box when it has risen by 4m. The student sketches a circle to represent the pulley, along with an overlapping square to indicate the pulley's hinge. A square represents the box, while line segments represent the wire going over the pulley. From the given information, the student determines that the work done on the box is $W = 480J$.

To verify the answer, the student decides to animate the sketch with the solution as input. Upon analysis, the pulley system and box are replaced with corrected diagram components (wires, pulley, hinge and square). The dotted lines are interpreted by the system to indicate the start and endpoints of movement. The direction of the arrow helps to distinguish the starting level from the ending level. The label for the arrow ($dx = 4$) is interpreted to indicate that the distance between the two dotted lines is to be taken as 4m. The box is assigned the label 'b', and the equation indicating mass ($m_b = 10$) of the box is associated implicitly. The student selects the equation pertaining to the solved answer ($w = 480$) by using the 'Lasso' gesture and completes the association by tapping the square. After this, the student triggers the animation.

If the computed value is correct, the box will move exactly 4m. If it is incorrect, then the box will come to rest either before or after reaching the end level. An incorrect animation will therefore provide the student with feedback about possible errors in the solution. Notice that the student worked out the answer as he would on paper, and only associated the minimum amount of required information. The equation for mass was associated implicitly. Additionally, the system was able to determine the beginning and endpoints of motion from the displacement indicator. No equations were written down to specify how the box's position should be updated per frame update.

Change in Gravitational Potential Energy During Free-Fall

Figure 8 represents a simple projectile problem. A ball of mass $m = 15$kg is dropped from a known height. The student is asked to determine the change in gravitational potential energy after the ball has fallen 50m. The student sketches a circle to represent the ball, and annotates the diagram with two horizontal dotted lines to indicate starting and ending levels for motion. An arrow is sketched between the two levels to indicate direction of displacement. From the given information, the student determines that the change in potential energy of the ball is $P_e = 7500J$.

To verify the answer, the student again decides to animate the sketch with the solution as input. Upon analysis, the ball is replaced by a circle. The dotted lines are interpreted by the

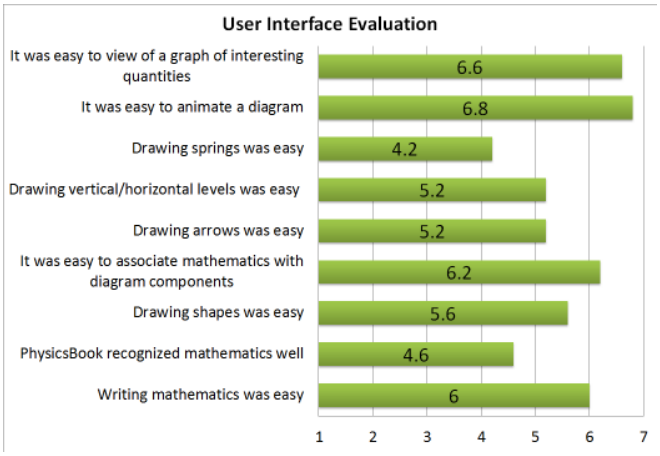


Figure 9. Mean participant ratings of individual aspects of PhysicsBook's user interface. Each of these questions required participants to respond using a 7-point Likert scale, where 1 was the most negative answer and 7 was the most positive answer.

system to indicate the starting and ending points of movement. The direction of the arrow distinguishes the starting level from the ending level, while also indicating the direction of motion. The label for the arrow ($dx = 50$) is interpreted to indicate that the distance between the two dotted lines is to be taken as 50m. The circle is assigned the label 'b', and the equation indicating mass ($m_b = 15$) is associated implicitly. The student selects the equation for potential energy ($P_e = 7500$) with the 'Lasso' gesture, associates it by tapping the circle, and triggers the animation.

The system will use the association to derive the magnitude of displacement and by utilizing the arrow annotation that depicts the direction of motion, will be able to construct the required displacement vector. If the derived change in potential energy is correct, the ball will move exactly 50m. If it is incorrect, then the ball will come to rest either before or after reaching the end level.

USER EVALUATION

We conducted a small-scale user study to get preliminary feedback about the perceived usefulness of PhysicsBook. Our goal in this study was to allow participants to experience the entire workflow of PhysicsBook. We chose two example scenarios that exposed participants to the potential of PhysicsBook.

Subjects and Apparatus

We recruited five participants (2 female and 3 male) from the University of Central Florida for an informal evaluation of PhysicsBook. The participants ages were between 22 and 28 years. Three participants had studied physics at the university level. Each participant took 15 to 20 minutes to complete the experiment tasks. The experiment was conducted on a HP Compaq TC4400 tablet computer equipped with an Intel Core 2 T5600 processor and 2 gigabytes of memory. The screen resolution was set to 1024x768 pixels. The tablet was placed on a table for the experiment. Participants sat down and used the stylus to interact with the tablet.

Experiment Procedure

Upon arriving for the experiment, each participant was given an introduction to PhysicsBook. This included instructions on how to draw shapes, springs, wires and pulleys. Participants were shown how to use the different annotations and how to associate mathematical expressions with recognized components of the diagram. They were also shown how to delete parts of the diagram by using the 'Scribble-Erase' gesture. Lastly, they were given an overview of the graphing functionality built into PhysicsBook.

For the experiment itself, each participant was given the solutions to two physics problems. The participant had to write out the given solution using PhysicsBook. The problems chosen for the experiment are shown in Figures 1 and 8. As PhysicsBook is not robust enough for a full user evaluation, we chose this controlled mode of interaction to get a preliminary idea of the effectiveness of the usage model for PhysicsBook and to measure the perceived usefulness of different components of the system. After the experiment, participants were asked to fill out a questionnaire that required them to rate PhysicsBook on a variety of metrics such as perceived difficulty in drawing the various diagram components, annotations, writing mathematics and making associations.

Results

Figure 9 shows participants' ratings of individual aspects of PhysicsBook's user interface. On the whole, participants responses seemed to indicate that they were able to draw diagram components and annotations in an easy fashion. The two notable exceptions are with respect to drawing springs and the ability of PhysicsBook to recognize mathematics well. The perceived difficulty of drawing strings is due to the fact that springs are easy to confuse with the 'Scribble-Erase' gesture in PhysicsBook. This led to poor performance and frustration among participants while drawing springs. As for the ability of PhysicsBook to correctly recognize handwritten mathematics, we suspect that the poor rating is due to our design choice where we hide the realtime recognition results for recognized mathematics. This approach was adopted in line with the findings of [23] who established that users prefer to trigger recognition once they complete an entire sketch. Our preliminary study seems to indicate that while this is true for the diagram part of the solution in physics problem solving, the participants in our study preferred to get realtime feedback about the handwritten mathematics. We plan to further investigate this aspect of user experience in the future with a larger set of participants. Figure 10 shows participants impressions of PhysicsBook's overall usefulness. It shows that despite the early stage of the prototype, participants felt positively about the potential usefulness of PhysicsBook.

DISCUSSION AND FUTURE WORK

Currently, the capability of PhysicsBook to animate a given problem is limited to select cases in domains related to classical mechanics. While we can associate and utilize mathematics written by a user, the system has limitations in sketch understanding and animation. From an animation perspective, it is important to know the starting and ending points of motion for each shape. Currently, we attempt to infer these points

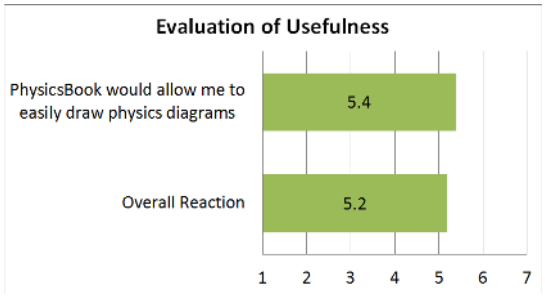


Figure 10. Mean participant rating of PhysicsBook's overall perceived usefulness. Both of these questions required participants to respond using a 7-point Likert scale, where 1 was the most negative answer and 7 was the most positive answer.

by the use of annotations and spatial reasoning. In problems related to potential and kinetic energy, we have demonstrated the usefulness of dotted lines combined with arrows to delineate displacement levels. However, the accurate determination of the beginning and end of motion in different contexts will be needed to expand the animation capability of our system. To this end, we plan to construct a database of solved problems whose analysis will yield insight into useful new annotations for sketch understanding. Such a database will also be useful for the discovery of inference rules for improvement of our inference module. Any new annotations and inference rules will enrich current functionality in terms of animation and sketch understanding, while allowing us to expand the range of problems that can be animated.

Most problems in physics require students to compute the answer in the form of a quantity at some point of interest. For example, a momentum problem may require students to compute momentum of one or more objects after an elastic collision. It is difficult to determine such points of interest in our current implementation. We can only use horizontal/vertical dotted lines to infer starting or ending points of motion. It is also not clear if such understanding can be derived from just the diagram and its annotations. We plan to utilize natural language processing techniques in the future to extract information from the problem statement itself, in order to achieve a better understanding of the solution.

Lastly, we use a customized monolithic physics engine to do animation. In its present form, it can only accept force, acceleration, mass, velocity and position as direct inputs. By using data transformations, we have expanded its capability to include concepts such as work, energy and momentum. It is unclear how much further we can extend its capabilities by using data transformations. In any case, the use of a monolithic animation system is limiting and bound to increase system complexity as the domain of diagrams becomes larger. In the future, we may utilize a framework that contains a series of animation modules tailored to different problem categories. Such an architecture will allow us to expand the range of physics concepts which can be animated significantly while keeping the design of each animation module manageable.

In addition to the constraints outlined above, we were forced to ask ourselves some interesting questions during the development of PhysicsBook.

1. What is a sufficient set of primitives and annotations that can be used to model a large set of problems in physics?
2. In some cases, the animation proceeds very quickly. Is there a practical lower limit on the duration of animation that impacts its usefulness in terms of facilitating learning?
3. If such a lower limit exists, how can we manipulate animations with durations below the limit to ensure their usefulness?
4. Is it possible to animate any given problem in a physics textbook? We believe that in reality, many physics problems can be animated with the proper software support but there may be an equally large number of problems for which the proper software support mechanism is unknown. Determining how to animate such problems is a challenging research question.

CONCLUSION

We have presented PhysicsBook, a prototype system that can recognize and animate sketched physics diagrams. PhysicsBook uses a customized physics engine for encoding domain knowledge that allows it to develop an understanding of physics problems that require a student to work out force(s), acceleration, velocity, displacement, position, or mass. Users can associate their own equations with the diagram by using a simple gesture set. In order to extend the capabilities of the animation system to branches of physics related to $f = ma$, we have designed a framework that can perform the necessary data transformations required to convert given variables to one of the acceptable inputs for our animation system. Use of data transformations in this fashion overcomes a limitation with previous approaches where users were forced to specify all aspects of animation mathematically. Our preliminary usability evaluation indicates a strong interest in the use of animations involving solutions to physics problems. Although there is more work required to reach our goals for a sketch-based physics tutoring system, we believe the PhysicsBook prototype is a solid foundation toward providing natural interfaces for physics understanding.

ACKNOWLEDGMENTS

This work is supported in part by NSF CAREER award IIS-0845921 and NSF awards IIS-0856045 and CCF-1012056. We would also like to thank the members of the ISUE lab for their support and the anonymous reviewers for their useful comments and feedback.

REFERENCES

1. Algodoo, 2011. `http://www.algodoo.com/wiki/Home`.
2. Alvarado, C. A natural sketching environmant: Bringing the computer into early stages of mechanical design. Master's thesis, MIT, 2000.
3. Cheema, S., and LaViola, Jr., J. J. Applying mathematical sketching to sketch-based physics tutoring software. In *Proceedings of the 10th international*

conference on Smart graphics, SG'10, Springer-Verlag (Berlin, Heidelberg, 2010), 13–24.

4. Cheema, S., and LaViola, Jr., J. J. Towards intelligent motion inferencing in mathematical sketching. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, ACM (New York, NY, USA, 2010), 289–292.
5. Forbus, K., Usher, J., Lovett, A., Lockwood, K., and Wetzel, J. Cogsketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science* (2011).
6. Forsberg, A., Dieterich, M., and Zeleznik, R. The music notepad. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, ACM (New York, NY, USA, 1998), 203–210.
7. Hammond, T., and Davis, R. Ladder, a sketching language for user interface developers. *Computers and Graphics 29*, 4 (2005), 518 – 532.
8. Igarashi, T., Matsuoka, S., and Tanaka, H. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co. (New York, NY, USA, 1999), 409–416.
9. Jiang, Y., Tian, F., Wang, H., Zhang, X., Wang, X., and Dai, G. Intelligent understanding of handwritten geometry theorem proving. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, ACM (New York, NY, USA, 2010), 119–128.
10. Jr., J. J. L. Advances in mathematical sketching: Moving toward the paradigm's full potential. *Computer Graphics and Applications, IEEE 27*, 1 (2007), 38 –48.
11. Kara, L. B., Gennari, L., and Stahovich, T. F. A sketch-based tool for analyzing vibratory mechanical systems. *Journal of Mechanical Design 130*, 10 (2008), 101101.
12. Laviola, Jr., J. J. *Mathematical sketching: a new approach to creating and exploring dynamic illustrations*. PhD thesis, Providence, RI, USA, 2005. AAI3174634.
13. LaViola, Jr., J. J., and Zeleznik, R. C. Mathpad2: a system for the creation and exploration of mathematical sketches. *ACM Trans. Graph. 23* (August 2004), 432–440.
14. Lee, W., de Silva, R., Peterson, E. J., Calfee, R. C., and Stahovich, T. F. Newton's pen: a pen-based tutoring system for statics. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, ACM (New York, NY, USA, 2007), 59–66.
15. Millington, I. *Game Physics Engine Development (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.
16. Newman, M. W., Lin, J., Hong, J. I., and Landay, J. A. Denim: an informal web site design tool inspired by observations of practice. *Hum.-Comput. Interact. 18* (September 2003), 259–324.
17. Oltmans, M., and Davis, R. Naturally conveyed explanations of device behavior. In *Proceedings of the 2001 workshop on Perceptive user interfaces*, PUI '01, ACM (New York, NY, USA, 2001), 1–8.
18. Ouyang, T. Y., and Davis, R. Chemink: a natural real-time recognition system for chemical drawings. In *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI '11, ACM (New York, NY, USA, 2011), 267–276.
19. Patel, R., Plimmer, B., Grundy, J., and Ihaka, R. Ink features for diagram recognition. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, ACM (New York, NY, USA, 2007), 131–138.
20. Paulson, B., and Hammond, T. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, ACM (New York, NY, USA, 2008), 1–10.
21. Purcell, E. J. E. J. *Calculus with analytic geometry / Edwin J. Purcell*, 3d ed ed. Addison-Wesley, 1978.
22. Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. The andes physics tutoring system: Lessons learned. *Int. J. Artif. Intell. Ed. 15* (August 2005), 147–204.
23. Wais, P., Wolin, A., and Alvarado, C. Designing a sketch recognition front-end: user perception of interface elements. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, SBIM '07, ACM (New York, NY, USA, 2007), 99–106.
24. Woolf, B. P. *Building Intelligent Interactive Tutors*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009.
25. Working model 2d, 2011. `http://www.design-simulation.com/wm2d/index.php`.
26. Xiong, Y., and LaViola Jr., J. J. Technical section: A shortstraw-based algorithm for corner finding in sketch-based interfaces. *Computers and Graphics 34* (October 2010), 513–527.
27. Young, H. D., and Freedman, R. A. *University Physics with Modern Physics*, 12th ed ed. Englewood Cliffs, N.J. : Prentice-Hall, 2008.
28. Zeleznik, R., Miller, T., Li, C., and Laviola, Jr., J. J. Mathpaper: Mathematical sketching with fluid support for interactive computation. In *Proceedings of the 9th international symposium on Smart Graphics*, SG '08, Springer-Verlag (Berlin, Heidelberg, 2008), 20–32.