AnimaChaotic: AI-based Automatic Conversion of Children's Stories to Animated 3D Videos

Reem Abdel-Salam, Reem Gody, Mariam Maher, Hagar Hosny and Ahmed S. Kaseb
Computer Engineering Department, Faculty of Engineering, Cairo University, Giza, Egypt

Keywords: Multimedia, AI, NLP, Storytelling, Scene Generation, Story Generation, Information Extraction.

Abstract: Stories are an effective and entertaining way of teaching children about real-life experiences in an engaging way. Although many children's stories are supplemented with graphical illustrations, having animated 3D video illustrations can effectively boost the learning process, especially for visual learners. However, producing animated 3D videos is a hard, expensive, and time-consuming process, so there is a need to automate this process. In this paper, we introduce AnimaChaotic, a story visualization system designed to automatically convert children's short stories to animated 3D videos by leveraging Artificial Intelligence (AI) and computer graphics. Our Natural Language Processing (NLP) pipeline extracts visualizable information from the story such as actors and actions. Then, our object positioning algorithm determines the initial positions of the objects in the scene. Finally, the system animates the scene using different techniques including AI behaviors. A quantitative analysis of our system demonstrates a high precision and recall in extracting visualizable information. It also shows that our system outperforms existing solutions in terms of static scene generation. A qualitative analysis of the system shows that its output is visually acceptable and outperforms similar solutions.

1 INTRODUCTION

People have used stories for thousands of years. Stories capture the attention of young children, teach them about the world around them, and summarize life experiences in a simple and engaging method. Most of the available children stories are text-based and sometimes supplemented with graphical illustrations to ease their understanding. This might not be enough for some children who would benefit more from a visual learning experience in which they can read, watch, and listen at the same time. Since reading is an important skill to learn at an early age, it is useful to have the stories supplemented with 3D video illustrations to encourage all the children to read while satisfying their different needs.

However, video illustrations are not available for all stories as they are hard, expensive, and time-consuming to produce. This is because producing animated 3D videos involves collecting or creating necessary models for actors and objects, constructing scenes relevant to the story, and developing the interactions of actors and objects in the scene. All this requires expertise (from artists and developers) in using advanced software tools. That is why there is a need for a system to automate the process of converting stories to 3D video illustrations and make it accessible to everyone

This paper proposes AnimaChaotic[1], a system that automatically converts children's short stories to animated 3D videos by leveraging AI and computer graphics. For an input story text, the system uses NLP techniques to extract visualizable information such as: i) Physical objects, their properties, and their constraints. ii) Actions performed by the story actors. iii) Dialogues between the actors. iv) Emotions of the actors. v) Weather conditions. vi) Background scene. Then, accordingly, the system produces the animated 3D video by constructing the scene, positioning the objects, scheduling the events, animating the scene, and overlaying the audio of the dialogues.

Experiments show that the precision and recall of extracting visualizable information is more than 96% (even for the most challenging information such as indirect objects). Then, we show that our system outperforms a similar existing solution in an end-to-end comparison in terms of static scene generation. Finally, we conduct a qualitative evaluation of the system using a user survey which shows that the output

[1]AnimaChaotic Video Trailers and Showcases:
http://www.youtube.com/watch?v=o8vY0U4bX1k
http://www.youtube.com/watch?v=Jz1qfO7PtAA

Abdel-Salam, R., Gody, R., Maher, M., Hosny, H. and S. Kaseb, A.
AnimaChaotic: AI-based Automatic Conversion of Children's Stories to Animated 3D Videos.
DOI: 10.5220/0010815100003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 2*, pages 226-235
ISBN: 978-989-758-547-0; ISSN: 2184-433X

from our system is visually acceptable and outperforms the competition.

The contributions of this paper are:

- To the best of our knowledge, AnimaChaotic is the first system to convert stories to animated 3D videos while supporting a wide range of visualizable information such as scenes, objects, actors, actions, emotions, dialogues, and weather conditions. The system is extendible such that new scenes, objects, actors, and emotions can easily be supported by extending the database.
- The system handles word and verb disambiguation to extract visualizable actions or events.
- We develop a robust graph-based object positioning algorithm in order to handle constraints between the scene objects.
- We support dynamic actions by defining action schemas (preconditions, execution mechanisms, and termination conditions). We use steering behaviors to implement navigation of the actors in the scene.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 details our methodology. Section 4 evaluates the system using a variety of techniques. Finally, Section 5 concludes the paper and summarizes future work.

2 RELATED WORK

Both NLP and Computer Graphics have recently seen rapid progress, but their integration when it comes to story visualization has not been extensively studied. Our work resides in this common intersection.

Information extraction is a crucial part of story visualization systems in order to extract visualizable elements from the story such as actors, actions, etc. ClausIE (Corro and Gemulla, 2013) and OLLIE (Mausam et al., 2012) are open information extraction systems that are used to extract triplets (subjects, verbs, and objects) from text. However, these systems do not generalize well to story text as they have been developed using factual texts such as Wikipedia. He et al. (He et al., 2017) use semantic role labeling for information extraction, but this system is trained on a news dataset, a far domain from stories. This urged the need to build our own information extraction system that is especially designed to handle stories.

Several similar solutions (Ma, 2002; Moens et al., 2015; Marti et al., 2018; Gupta et al., 2018) have been proposed to convert stories into animated videos. However, our system is different from them in several ways. CONFUCIUS (Ma, 2002) supports actors and actions only, but it focuses more on language to humanoid animation. MUSE (Moens et al., 2015) is limited to a pre-determined graphical scene and does not focus on scene generation. CARDINAL (Marti et al., 2018) requires the input text to follow the standardized format of movie scripts, and it only considers subjects, verbs, and objects. Gupta et al. (Gupta et al., 2018) developed a retrieval storytelling system in which the system retrieves a short video from a database to represent each sentence in the input story. These short videos are then processed and concatenated to produce the final video. One downside of this approach is being limited by the video frames stored in the database and the nature of the input stories.

Other solutions have been proposed to convert text into static 3D scenes or images using a variety of techniques. These solutions include WordsEye (Coyne and Sproat, 2001), Text2Scene (Tan et al., 2019), the system proposed by Lee et al. (Lee et al., 2018), SceneSeer (Chang et al., 2017), and the system introduced by Chang et al. (Chang et al., 2014) which focuses on spatial relations between objects and inferring missing objects in the scene. These systems are limited to static scene generation, i.e., not dynamic or animated as in the case of our system.

3 METHODOLOGY

Figure 1 shows the architecture of our proposed system which is divided into an NLP pipeline and a graphics pipeline. The story is entered as text. It is then processed by the NLP pipeline which extracts all the visualizable information. This information is structured in a special format and fed into the graphics pipeline. This pipeline handles loading the scene, positioning objects in the scene, and applying the appropriate animations. The following subsections explain the components of each pipeline in detail.

3.1 NLP Pipeline

This pipeline extracts the information that can be visualized and animated from the story text. This information represents objects, actors, positioning constraints, and events. Objects can have characteristics such as color and shape. Actors can be described by their gender, age, height, physical appearance, and clothes. Positioning constraints define the spatial relations between objects and actors in the scene. Events can be changes in the weather conditions, changes in the emotions of the actors, or actions performed by

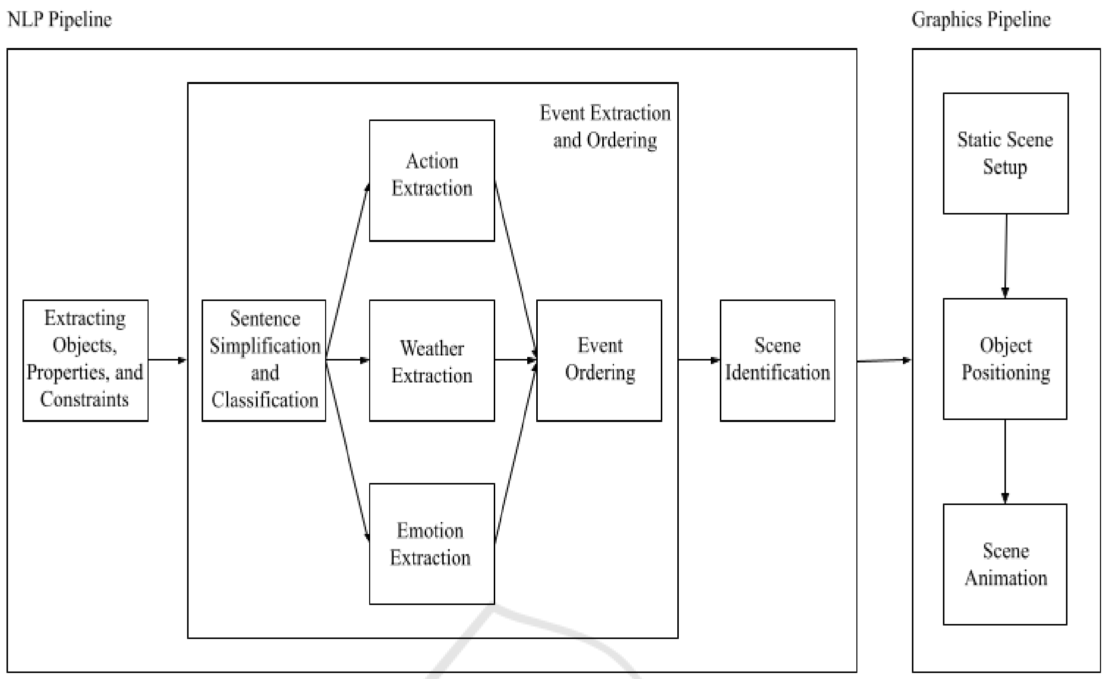


Figure 1: System Architecture.

the actors (sequentially or in parallel). The system supports simple sentences as well as some forms of complex sentences as explained later. In addition, it is assumed that the whole story takes place in the same location (i.e., same scene with no scene transitions).

3.1.1 Extracting Objects, Properties, and Constraints

Using the input text, the system extracts the physical objects, their properties, and their constraints:

- Extract the objects by first finding all the nouns using a Part-Of-Speech (POS) tagger. Then, among these nouns, physical objects are identified using the WordNet database (Miller, 1995).
- Identify the properties of each object. These types of properties are supported: count, color, hair and eye color (for humans), shape (e.g., tall, short, thin, and fat). This is achieved by identifying the noun phrase of each noun (if exists), then extracting any existing properties from the noun phrase.
- Discover the constraints between the objects by identifying the prepositions (e.g., on, below, and behind) and their relevant objects.

3.1.2 Event Extraction and Ordering

Sentence Simplification and Classification: This is responsible for converting complex and compound sentences into simple sentences. We require that the sentence has either one or two events only. This requirement was largely put in place due to ambiguity introduced when a sentence contains three or more events. Consider for example: “John ate a burger after Sam hugged the kitten before the kitten purred.” In this example, it is ambiguous whether John ate the burger before the kitten purred or whether Sam hugged the kitten before the kitten purred. There is a similar ambiguity at the “after” keyword. Allowing only simple sentences with fewer than three events removes this source of ambiguity.

Then, we classify sentences based on their main verbs into action and non-action sentences using VerbNet (Schuler and Palmer, 2005). Verbs can be action or non-action depending on their context, so we can not rely solely on VerbNet, especially when using verbs describing actions in progress such as start, stop, and begin. For example, *start* is a non-action verb, but the sentence “He started to play football.” is an action sentence since *start* implies the beginning of an action. Action sentences are then further processed by the Action Extraction component as explained later. Non-action sentences are then further processed by the Weather Extraction and Emotion Extraction components as explained later.

Action Extraction: Using the input text, the system extracts all the actions that can be visualized and identifies all their relevant information. This is

Table 1: Action Extraction: The information that can be extracted for a single action, how this information can be found in the text, and some simple real-life examples. Prepositional objects are classified into different categories (e.g., location and physical object) using a named entity recognition model and a knowledge base.

Extracted Information	**Representing Rule**	**Example**
Action	Verb	The boy **ran** to his mother.
Actors performing the action	Subject	The **boy** ran to his mother.
	Prepositional object representing an actor	The boy ran with his **sister** to his mother.
Objects or actors affected by the action	Object	The boy was eating a **sandwich**.
	Indirect object	The boy gave his mother a **flower**.
Positional constraints of the actors during the action	Prepositional object of a location-based preposition	The boy was standing **on the bed**.
Destination or direction of the action	Prepositional object of a motion verb	The boy ran to his **mother**.
Emotions of actors during the action	Adjective	The **happy** boy was playing.
	Adverb	The boy was playing **happily**.
	Prepositional object representing emotions	The boy was playing with **joy**.
Instruments used to perform the action	Prepositional object representing a physical object	The boy was playing with the **ball**.

achieved by first using a POS tagger to assign tags (e.g., noun, verb, or conjunction) to all the individual words. Then, non-action verbs that can not be visualized (e.g., breathe and smell) are identified using VerbNet in order to be ignored in the visualization process. Sentences with action verbs, on the other side, are further processed to identify all the relevant information about their actions. Table 1 lists the information that can be extracted for a single action, how this information can be found in the text, and some simple real-life examples. Compound subjects, objects, and verbs are supported as well. A compound subject consists of multiple subjects performing the same action (e.g., Mary, John, and Alice were playing with the ball.). A compound object consists of multiple objects being acted upon (e.g., Mary was carrying an apple and an orange.). The same applies to the compound verb which represents multiple actions performed by the subject. Most rule-based information extraction systems that we tested (ClausIE and OLLIE) do not handle these cases, but our system handles this by following conjunctions to find the actual POS using the sentence's dependency parse tree.

Complex sentences are also supported. For example, consider this sentence: "John and Mary were playing happily with the ball, in front of Sally, in the afternoon." This information is extracted:

- **Action**: play
- **Actors**: John and Mary
- **Affected Object**: ball
- **Positional Constraints**: in front of Sally
- **Emotions of Actors**: happily

A special kind of action is when characters communicate with each other in the story. For this, we use some techniques to detect the conversations, and the speakers are detected as explained earlier. Quotes (when they exist) are used as a strong feature to detect the speech segments. In the absence of quotes, we develop some rules that are based on the parse tree of the sentence. Verbs implying speech (e.g., say and shout) and the first succeeding non-speech verb (if exists) are detected. Afterwards, the sentence is split at the subject associated with the non-speech verb. For example, in the sentence "John said that the weather is fine.", the sentence is split at "the weather" and "the weather is fine" is used as the extracted speech segment. In case that a non-speech verb appears before the speech verb, the split is done at either the speech verb if its subject appears afterwards (e.g., The weather is fine, said John.), or at the speech verb's subject if the subject appears before the verb (e.g., The weather is fine, John noted.).

Verbs can have different meanings based on their context. For example, *run* at "John was running the factory." means manage, but it implies motion at "John was running with his sister.". Another example is that *jump* is an action verb at "John was jumping.", but it is a non-action verb at "The club was jumping with music.". To handle these cases, we develop some linguistic rules for common verbs to handle their different meanings. For example, *run* could only imply motion in case it occurs as an intransitive verb.

Weather Extraction: From the input text, the system detects weather-related words using WordNet.

For each detected word (and all its synonyms), the system queries a knowledge base to categorize the weather conditions into one of the following supported categories: sunny, windy, rainy, snowy, foggy, and cloudy. The system detects weather conditions whether they apply to the entire story (e.g., It was a sunny day.) or they occur in the middle of the story (e.g., Then, it suddenly rained.).

Emotion Extraction: Some emotions are not related to a certain action (e.g., He got angry.), hence they are not handled during action extraction. These emotions are extracted as follows: i) Use a POS tagger to extract nouns, adverbs, and adjectives. ii) Extract the synonyms of these words using WordNet. iii) Determine if each word or any of its synonyms is emotion-related or not using a knowledge base. iv) Categorize each word into six categories using an emotion lexicon: fear, joy, anger, surprise, disgust, and sadness.

Event Ordering: After extracting events (weather change, emotion change, and actions) from the story, we need to determine their chronological order. Normally, we assume that the story events are sequential unless special keywords appear. We use keywords such as "then" and "after" to determine the correct sequential order of events. Additionally, keywords such as "while", "when" and "at the same time" are used to determine action parallelism. For compound sentences, if the actors in the sentence are different, their actions are considered as parallel actions. An example of these types of sentences is: "John was playing football in the garden, and Mary was singing.". If the sentence has one actor doing multiple actions, these actions are considered sequential.

3.1.3 Scene Identification

The system identifies the general scene of the story among the following currently supported scenes: house, street, farm, beach, and garden (could easily support more in the future). The scene can be explicitly mentioned in the story or implied by the objects and actions. The system identifies the scene in both cases. If it is explicitly mentioned, the system uses rule-based techniques to identify the scene and exclude scenes mentioned in speech sentences or negated sentences. If it is not explicitly mentioned, the probability of each scene is calculated based on the objects and the actions found in the story, returning the most likely scene. We use the Naive Bayes classifier 1 to find the probability of each scene given the probabilities of having these objects and actions in this particular scene as follows:

$$\underset{k}{\operatorname{argmax}} P(S_k|Story) = \underset{k}{\operatorname{argmax}} P(S_k) \prod_{i=1}^{N} P(Word_i|S_k) \tag{1}$$

where S_k is the scene k, $P(S_k|Story)$ is the probability of scene k given the words in the story, $P(S_k)$ is the probability of each scene k, which is assumed to be uniform for all scenes, $P(Word_i|S_k)$ is the conditional probability of the occurrence of each word in a given scene k (built by scrapping the Internet to find the possibility of having different objects and actions in different scenes).

3.2 Graphics Pipeline

To achieve modularity and to allow integration of more advanced NLP or graphics pipelines in the future, we develop a flexible and extendible representation that captures all the details in the input story. This representation is passed from the NLP pipeline to the graphics pipeline, and it includes:

- Identified scene, e.g., garden or room.
- Weather information, e.g., sunny or rainy.
- Time of the day, e.g., day or night.
- Static objects (e.g., cup), their counts and colors.
- Actors (i.e., objects that can move, e.g., boy or dog) and their hair and eye colors.
- Positional constraints (e.g., beside or above) between the objects and the actors in the scene.
- Actions where each action can be:
  - Single: An action performed by a single actor.
  - Aggregate: An action performed by a group of actors together, e.g., dancing. The verb associated with each action (single or aggregate) can be: a verb with no objects (e.g., run), a verb with direct objects, or a verb with direct and indirect objects.
  - Event: Represents changes in weather conditions (e.g., It suddenly rained.) or actors' emotions (e.g., He became sad.).
  - Parallel: A set of actions (single, aggregate, or event) that are happening at the same time. For example, the boy was running while the girl was eating.

The Graphics module is responsible for rendering the scene video in a way that captures all this information. It achieves this as follows:

1. Load a terrain based on the scene location. For example, a grassy terrain should be used if the location is a garden.

2. Load a background depending on the scene location, time, and weather details.
3. Generate the static scene:
   - Select actors such that they satisfy their genders, ages, and properties (e.g., hair color).
   - Use our proposed object positioning algorithm to position the objects and the actors in the scene such that they satisfy their initial positional constraints.
4. Animate the scene by moving the actors around:
   - Select the appropriate animations based on the given actions such as walking and running.
   - Use AI behaviors (e.g., moving to a point while avoiding obstacles) to execute the actions.

3.2.1 Static Scene Setup

The graphics pipeline starts by retrieving the main scene elements from the database in order to setup the static scene. Our database contains a set of background images for each scene type satisfying different times of the day such as morning and evening. It also contains images for different terrains that are used as textures for the ground plane. In addition to this, objects and actors are stored in the database along with metadata that aids in their retrieval such as the type of the object and the gender of the actor. An appropriate scene background is chosen such that it satisfies the identified scene and the time information. A suitable terrain image is then selected according to the scene type. We retrieve the objects from the database based on their type and count, and we retrieve the actors such that they satisfy their genders, ages and properties. The actors and objects are then positioned in the scene according to our proposed algorithm that is described in the next subsection.

3.2.2 Object Positioning

The main objective of the object positioning algorithm is to place the input objects in the scene such that they are all visible and the positional constraints between them are satisfied. Constraints are either binary (i.e., between two objects, e.g., beside, above, inside, or behind) or non-binary (i.e., among a group of objects). The system handles both types by converting non-binary constraints to a set of binary constraints between pairs of objects in the group. Throughout our discussion, we will consider the x-axis goes to the right of the screen, the y-axis goes into the screen, and the z-axis is the upwards axis.

Algorithm 1 shows the details of the object positioning algorithm which works as follows:

1. Pre-process constraints by excluding unsupported ones and converting non-binary constraints to binary ones as explained above.
2. Place objects with no imposed constraints in random visible positions.
3. Build an undirected graph with the vertices representing the constrained objects and the edges representing the XY constraints between them.
4. Find the list of Strongly Connected Components (SCC) in the graph using Depth-First Search. Each SCC represents a group of objects with XY constraints among them. Objects are distributed in a 2D array that reflects these constraints (same row: beside, same column: in front or behind).
5. Build a directed graph with the nodes representing the SCCs and the edges representing the Z constraints. In the graph, a parent SCC should be positioned below the child SCC according to their Z constraints.
6. Perform topological sort on that directed graph in order to always have each parent SCC before its children. This allows the system to position the parent SCCs first then their children due to their Z positional dependency.
7. Position SCCs and objects:
   - Each SCC is treated as a single unit when positioned in the scene.
   - Position the SCC whether it is depending on an earlier SCC or not.
   - Within each SCC, position all objects while satisfying their XY constraints.
8. Perform any necessary scaling to the objects such that the objects placed inside other objects fit within them.

3.2.3 Scene Animation

Animating the scene is realized by executing the actions in the appropriate chronological order. Each action can either be: single, aggregate, event, or any of these in parallel. Single actions can have any of the following types: walk, wander, sit, drink, eat, jump, fall down, run, pick, say, exercise, punch, dance, give, and bark. These types cover a wide variety of visualizable actions that are extracted by the NLP module and can be easily extended as all of them share the same interface. Each action has a precondition, an execution mechanism, and a termination condition.

Algorithm 1: Object Positioning Algorithm.

```
Input: Objects and their positional
  constraints
Output: Objects' 3D coordinates

foreach constraint in constraints do
    if constraint is supported then
        Include constraint to be applied;
        Convert to binary constraint if
          applicable;
    else
        Exclude constraint;
    end
end
foreach object in unconstrained objects do
    Set X, Y, Z to a random visible location;
end
graph1 = Graph(nodes=objects,
  edges=XYconstraints);
SCC = findConnectedComponents(graph1);
graph2 = Graph(nodes=SCC,
  edges=Zconstraints);
orderedComponents =
  topologicalSort(graph2);
foreach component in orderedComponents
  do
    positionComponent(component);
end
foreach object in objects do
    scale(object, object constraints);
end
```

- The **precondition** is required to start executing the main action. For example: i) To give an object to an actor, a precondition would be to pick the object. ii) To pick an object, a precondition would be to walk towards the location in which the object exists in the scene. iii) To walk towards an object, the actor must be in a standing posture (not sitting on a chair for example).
- The **execution** of the action is mainly playing a representative animation until the termination condition is met. For actions such as walk, run, and wander, the execution requires the invocation of steering behaviors that ensure the smooth navigation of actors in the scene. For actions that imply speech such as say and talk, we use a third-party text-to-speech service to allow our actors to speak in voices that reflect their gender. We also display the speech text in speech bubbles above the actor. For actions such as bark and meow that are usually associated with animal actors, we play animal voices to make the story more lively.
- The **termination condition** marks the end of action execution by the actor. For example, for the pick action, the termination condition is that the actor has the object in his grip. For some actions such as wander, it may be difficult to have a clear termination condition. For this, we define the termination condition as a time interval after which the action should stop executing.

For aggregate actions that are being performed by multiple actors, the system replicates the single action for all the actors. Events are classified into weather change and emotion events. For weather change, *rainy* and *snowy* are simulated using particle effects, and *cloudy* is simulated by randomly moving cloud models. To support a wide variety of emotions in a way that is easily extendible and to ensure that they are supported for the different types of actors, we use the idea of emojis which are popular in chatting applications to express emotions. Emojis are loaded above the actor in order to indicate the desired emotion.

The navigation of the actors around the scene is achieved by implementing Reynolds' steering behaviors (Reynolds, 1999) which determine the navigation path while not making assumptions about the scene or the actors. These steering behaviors assume point mass approximation where an object is treated as a point with finite mass and negligible dimensions. Each object is characterized by a velocity that is modified by applying a force. Treating objects as points allows us to simplify the calculations and to handle the motion of a wide variety of models independent of their dimensions. Each steering behavior is implemented by calculating a vector representing the desired steering force. We implement three behaviors:

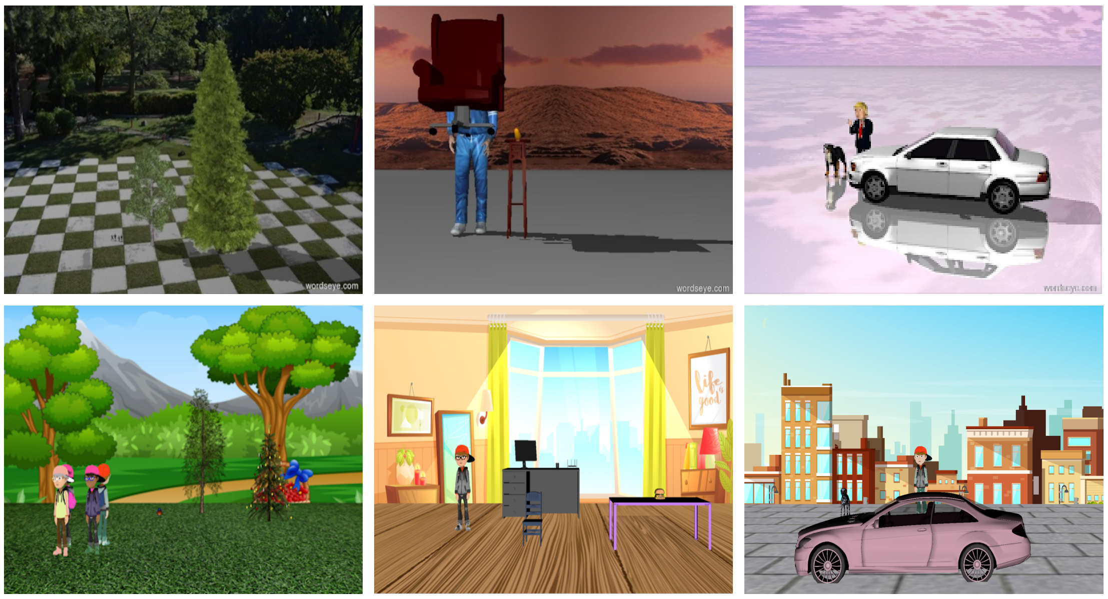
- **Seek** is concerned with moving towards a point.
- **Wander** is a type of random steering implemented by generating random forces to simulate the wandering behavior of the actors in the scene.
- **Obstacle Avoidance** is concerned with generating forces that allow the actor to steer away from nearby objects.

The steering behaviors are combined to generate more complex behaviors. For example, to allow an actor to move towards a target without hitting objects, we calculate the weighted average of the seek and obstacle avoidance forces.

3.2.4 Object Database and Graphics Tools

We collect a database of models from various online sources and use MongoDB[2] to store metadata about

[2]https://www.mongodb.com



(i) (ii) (iii)

Figure 2: Visual comparison between AnimaChaotic (bottom row) and WordsEye (top row) using different scenes. i) AnimaChaotic produces a clear view of all the objects compared to WordsEye in which the four children are very small. Text: "There were two tall trees. There was a bird sitting on the tree. There were four boys in the garden." ii) AnimaChaotic infers the scene (i.e., room) from its objects (e.g., table and chair) while WordsEye does not. Text: "The food was on the table. John has brown hair. The chair was in front of his computer." iii) AnimaChaotic infers the scene (i.e., street) from its objects (e.g., car) while WordsEye does not. Text: "A dog is next to the boy. A car is in front of the boy."

the objects. Mongo is a NoSQL document database. It has the advantage of being schema-less and thus we can save specific information about each object without adhering to the constraints that are usually imposed by structural databases. The metadata includes: the type of object, its dimensions, and the names of the animations it supports. This allows us to retrieve the objects accurately and scale them relative to each other. We handle the color property of the object by loading different textures when they exist. For supporting multiple animations, we use Mixamo[3] that allows us to fit a fully rigged skeleton to the humanoid models we have and generate different animations for each model. We also use Mixamo's free models to expand our database with more models. We support more than one model for each object category to allow our scenes to be more versatile. We use the Panda3D[4] game engine for rendering of the scene.

4 EVALUATION

This section evaluates the system in several ways: accuracy of NLP information extraction, the accuracy of scene generation and object positioning compared to WordsEye, end-to-end human evaluation, and comparison to similar solutions in terms of supported features. We choose WordsEye for our experiments because it is the only similar solution that allows users to try their system online.

[3]https://www.mixamo.com/

[4]https://www.panda3d.org

Table 2: Precision and recall of extracting different information from eleven stories.

Extracted Information	Precision	Recall
Scene	100%	100%
Actor	98.98%	100%
Emotion	100%	100%
Action	100%	98.41%
Direct Object	100%	97.17%
Indirect Object	100%	96.36%

NLP Information Extraction: Eleven simple stories were used to evaluate the system in terms of the precision and recall of the extraction of the scene, actors, emotions, actions, direct objects, and indirect objects. Table 2 presents the results which show that the system is both accurate and robust when tested with unseen stories.

Scene Generation and Object Positioning: Nineteen stories were used to evaluate the system against WordsEye. Those stories were selected to satisfy the requirements of WordsEye since it supports generating static scenes only, and does not visualize the dynamic actions. It also requires simple sentences that follow a certain structure. Figure 2 shows a visual comparison between our system and WordsEye using different scenes.

Figure 3 evaluates the two systems in terms of precision, recall, and F1 score of object extraction and constraint satisfaction. To calculate precision and recall, true positives are the objects or constraints that are mentioned in the story and visualized in the output. False positives are the objects or constraints that are incorrectly visualized in the output. False negatives are the objects or constraints that are mentioned in the story, but not visualized in the output. The figure shows that our system performs better in terms of object precision and object F1 score.

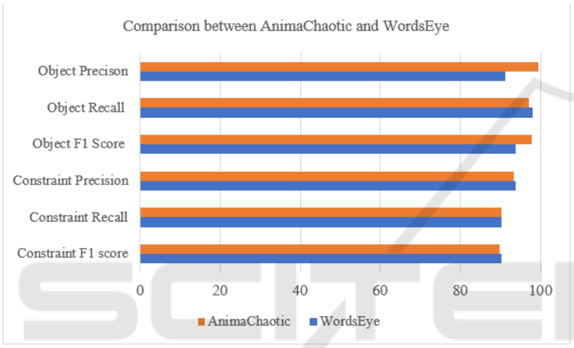


Figure 3: Precision, Recall and F1 scores for objects and constraints satisfaction for AnimaChaotic and WordsEye.

Table 3 compares the two systems in terms of scene choice accuracy for which our system greatly outperforms WordsEye due to the technique described in section 3.1.3 (using Naive Bayes to infer the scene based on its objects if the scene is not explicitly mentioned in the text). In addition, the table compares the two systems in terms of color satisfaction accuracy which means the ability of the system to satisfy the colors of the objects mentioned in the story. WordsEye performs better as it has a wider support for textures for different objects.

Human Evaluation: Evaluating visual information can be very subjective. That is why we also evaluate our system by human audience. Two surveys were used for evaluation: i) Figure 4 shows the results of the first survey which presents output images from 10 stories from our system and WordsEye. For each story, the survey asks “Which image is more accurate and visually acceptable?”. ii) Figure 5 shows the results of the second survey which presents output videos from 4 stories from our system. For each story, the survey asks “To which extent the output matches the text?”, and the possible answers are: Excellent, Very Good, Good, Fair, and Poor. The two surveys show that the output from our system is visually acceptable to the audience.

Table 3: Comparison between AnimaChaotic and WordsEye in terms of color accuracy and scene choice accuracy.

System	**Color Accuracy**	**Scene Choice Accuracy**
AnimaChaotic	91.25%	90%
WordsEye	94.38%	55%

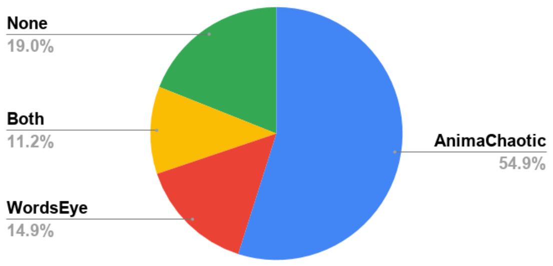


Figure 4: Human evaluation of AnimaChaotic’s output and WordsEye’s output.

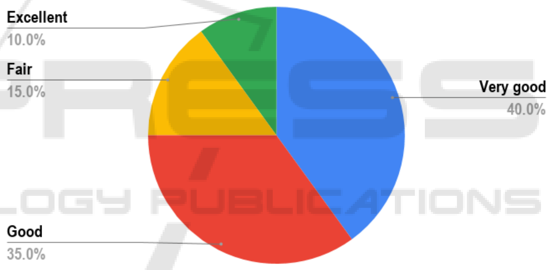


Figure 5: Human evaluation of AnimaChaotic’s generated videos.

Supported Features: Table 4 compares our system to the similar solutions in terms of supported features such as object extraction, scene identification, emotion extraction, etc. This shows that our system supports the most features.

5 CONCLUSION

This paper proposes AnimaChaotic, a system that generates video illustrations from childrens’ stories. AnimaChaotic leverages NLP and computer graphics and introduces a pipeline that extracts visualizable information from the story text and presents it in the form of an animated 3D video. AnimaChaotic was tested on a set of stories that we devised as well as adapted from actual visual stories on the Internet. Our experiments show that our proposed system is com-

Table 4: Comparison between AnimaChaotic and similar solutions in terms of supported features.

	AnimaChaotic	WordsEye	MUSE	CONFUCIUS
Object extraction	✓	✓	✓	✓
Scene identification	✓	✓	—	—
Animation	✓	—	✓	✓
Emotion extraction	✓	—	—	✓
Positioning constraints	✓	✓	—	—
Weather extraction	✓	—	—	✓
Interactive	—	✓	✓	—

petitive with existing solutions. A human evaluation of the system emphasizes that its output is visually acceptable and matches the text to a great extent. In the future, AnimaChaotic can be improved to take into account long term dependencies between sentences. AnimaChaotic can also be extended to support multiple scenes within the same story.

REFERENCES

Chang, A. X., Eric, M., Savva, M., and Manning, C. D. (2017). SceneSeer: 3D scene design with natural language. *ArXiv*, abs/1703.00050.

Chang, A. X., Savva, M., and Manning, C. D. (2014). Learning spatial knowledge for text to 3D scene generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Corro, L. D. and Gemulla, R. (2013). ClausIE: clause-based open information extraction. In *Proceedings of the International Conference on World Wide Web*.

Coyne, B. and Sproat, R. (2001). WordsEye: An automatic text-to-scene conversion system. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*.

Gupta, T., Schwenk, D., Farhadi, A., Hoiem, D., and Kembhavi, A. (2018). Imagine this! scripts to compositions to videos. In *Proceedings of the European Conference on Computer Vision*.

He, L., Lee, K., Lewis, M., and Zettlemoyer, L. (2017). Deep semantic role labeling: What works and what's next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Lee, D., Liu, S., Gu, J., Liu, M.-Y., Yang, M.-H., and Kautz, J. (2018). Context-aware synthesis and placement of object instances. In *Proceedings of the International Conference on Neural Information Processing Systems*.

Ma, M. E. (2002). Confucius: An intelligent multimedia storytelling interpretation and presentation system. *First Year Report, School of Computing and Intelligent Systems, University of Ulster, Magee*.

Marti, M., Vieli, J., Witoń, W., Sanghrajka, R., Inversini, D., Wotruba, D., Simo, I., Schriber, S., Kapadia, M., and Gross, M. (2018). Cardinal: Computer assisted authoring of movie scripts. In *Proceedings of the International Conference on Intelligent User Interfaces*.

Mausam, Schmitz, M., Soderland, S., Bart, R., and Etzioni, O. (2012). Open language learning for information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Moens, M.-F., Do Thi, N. Q., Kordjamshidi, P., and Ludwig, O. (2015). Visualising the content of a text in a virtual world. In *Kultur und Informatik: Cross Media*, pages 33–37. Verlag Werner Hülsbusch.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Game Developers Conference*.

Schuler, K. K. and Palmer, M. S. (2005). *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania.

Tan, F., Feng, S., and Ordonez, V. (2019). Text2Scene: Generating compositional scenes from textual descriptions. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.