5
Interactive Film Production

Current advances in interactive storytelling technology have been based mostly on the development of computational methods for the generation, interaction, and dramatization of interactive narratives – focusing mainly in the audience side of the storytelling process. However, in any form of storytelling, the author is the key component for a successful story. The concept of authoring in interactive storytelling has its origins with the own research field, but only recently it has attracted the attention of the research community (Medler and Magerko 2006; Spierling et al. 2006; Pizzi and Cavazza 2008; Riedl et al. 2008; Swartjes and Theune 2009). However, even today, there is a clear imbalance between the number of technical proof-of-concept prototypes and the number and scale of actual interactive narratives.

In an attempt to reduce the gap between interactive storytelling systems and film directors, this chapter presents a general guide on how to write and film interactive stories, and describes some computational tools developed for the production of video-based interactive narratives.

The process of production of a video-based interactive narrative is divided into three phases: (1) pre-production – where the story is logically defined by the author and the shooting script describing how to shoot the film elements is automatically generated; (2) production – where the film set is constructed and the raw elements (actors' actions and locations) are recorded; and (3) post-production – where the raw material is edited, the background is removed, and the video files are associated with the corresponding characters' actions and locations.

5.1.
Pre-production

The first step of the process of production of a video-based interactive narrative is the pre-production phase. It involves the logical definition of the story, including the definition of the characters, their attributes and relations, the

locations where the story take place, the possible events that can occur during the story and the way these events are represented.

5.1.1. Story Definition

Interactive stories are generated according to a story context, which comprises a logical description of the mini-world where the narrative takes place, a definition of the events that can be enacted by the characters, and a description of the motives that guide their behavior.

The process of creating the story context involves work from both story writers and programmers. Initially, the author of the story creates a draft of the general story idea, describing the diverse story elements (e.g. characters' psychology, representative scenes and environment description). Based on these drafts and using cognitive modelling and programming skills, the programmer describes the story world and the various events of the story according to their validity conditions and their consequences.

The elements that compose the story context can be divided in:

- **Static schema**: describes the mini-world wherein the plots take place (characters, relations, places…);
- **Dynamic schema**: describes the possible events in which the characters of the story can participate;
- **Behavioral schema**: describes the motives that guide the behavior of the characters;
- **Detailed schema**: details the description of the story events in terms of actions.

5.1.1.1. Static Schema

The formal specification of the static schema is based on the standard syntax and semantics of first order languages and follows the formalism of the Entity-Relationship model (Batini et al. 1992). The static schema is composed by a set of facts indicating the existence of entities, the values of the attributes of entities, the

existence of relationships, the values of the attributes of the relationships, or the assignment of roles to entities. An entity can represent anything of interest by itself, material or abstract, animate or not. A set of facts holding at a given instant of time constitutes a state.

The clause patterns used in the specification of the static schema are given below. In conformity with Prolog conventions, square brackets are used for conjunctive lists (with "," as separator) and round brackets for disjunctions (with ";" as separator).

```
entity(<entity-class>,<identifier>).
relationship(<relationship-class>,[<entity-class>,...,
                                          entity-class>]).
attribute(<entity-class>,<attribute>).
attribute(<relationship-class>,<attribute>).
boolean(<attribute>).
composite(<attribute>,[<attribute-part>,...,
                                      <attribute-part>]).
is_a(<more-specialized-entity-class>,
                                <more-general-entity-class>).
role(<role>,(<entity-class>;...;<entity-class>)).
```

In the context of a narrative, the major entity classes are usually the characters and places where the story takes place. The characters are identified by their names and may have some attributes describing their physical and emotional characteristics. Similarly, locations are also identified by a name and may have some attributes. Between characters and places there may be some relationships, for example, indicating the home or the current place of a character. Similarly, relationships between characters are also accepted, for example, indicating the affection of one character to another or indicating that two characters are married.

The static scheme is created according to the general idea of the story plot. For example, considering the following plot idea:

"*A young boy named Peter falls in love with a girl named Anne who he met at the university. Advised by two imaginary creatures, a little angel and little devil, Peter tries to know more about Anne by hacking her Facebook page. After getting some information, Peter manages to go out with Anne on a date after a*

party, but she finds out that he invaded her social network account. After so many lies, Anne does not know whether she forgives Peter."

There are four characters that participate in this story (Peter, Anne, Angel and Devil) and at least four different locations (university, party, Peter's house and Anne's house). Based on these assumptions, it is possible to identify five entity classes:

- creature – all the characters are living creatures;
- person – both Peter and Anne are persons;
- adviser – both Angel and Devil are advisers;
- student – both Peter and Anne are students;
- place – all the locations where the story happens are places.

Every student is a person and every person and adviser is a creature. A person has a gender and every living creature has a level of joy and affection for other creatures. The story has protagonists (students) and supporting characters (advisers). The complete entity-relationship diagram of these various components and the connections among them are shown in Figure 5.1.

Based on the definition of the entities and relationships, the static scheme can be specified in a concrete notation:

```
/* Static Schema */
entity(creature, name).
entity(person, name).
entity(adviser, name).
entity(student, name).
entity(place, place_name).
is_a(person, creature).
is_a(student, person).
is_a(adviser, creature).
attribute(person, gender).
attribute(creature, joy).
attribute(acquaintance, affection).
relationship(home,[creature, place]).
relationship(current_place, [creature, place]).
```

```
relationship(acquaintance,[creature,creature]).
relationship(know, [creature, creature]).
relationship(know_more, [creature, creature]).
relationship(seduced, [person, person]).
relationship(debunked,[creature, creature]).
relationship(forgiven,[person, person]).
role(protagonist, (student)).
role(supporting, (adviser)).
```

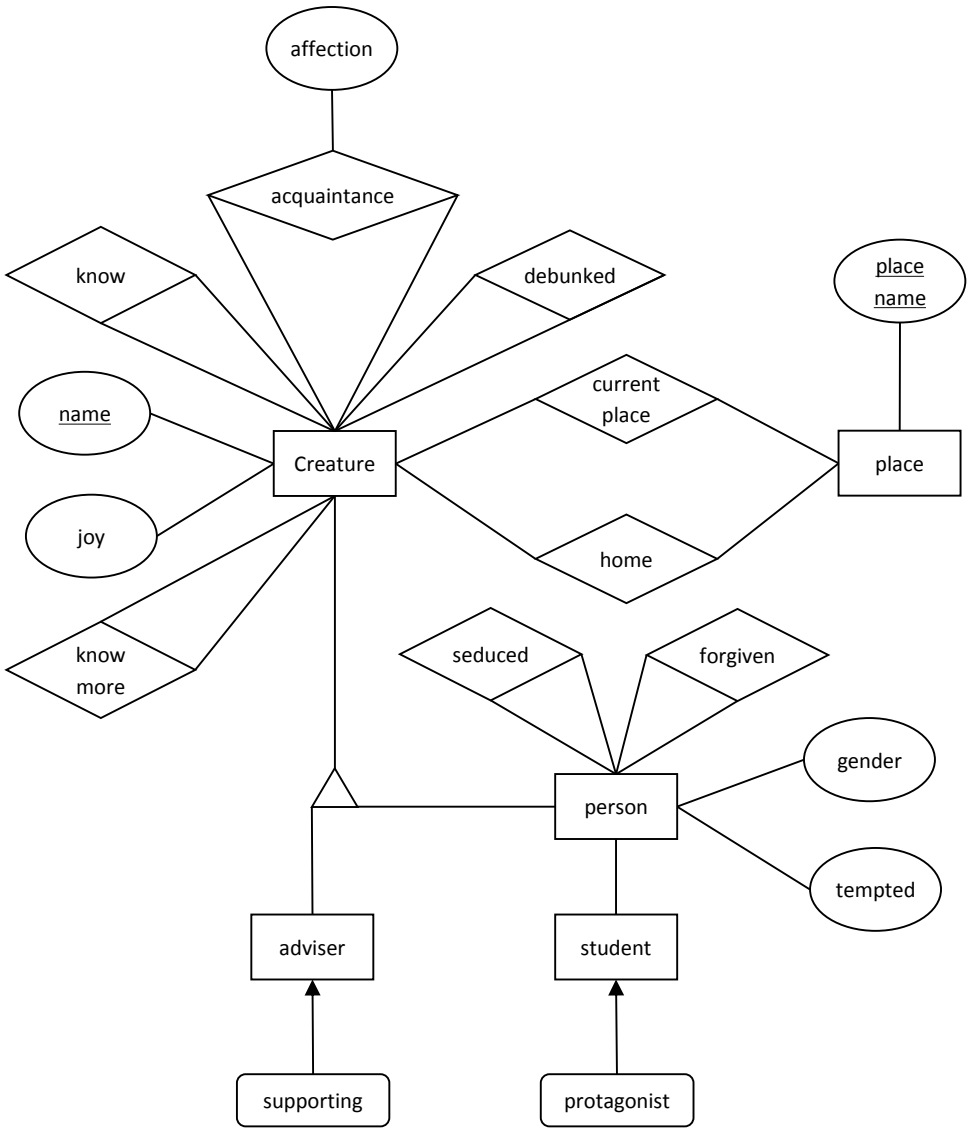

Figure 5.1: Entity-Relationship diagram of the static scheme.

The notation for representing facts is a straightforward consequence of the schema notation. The terms `db(<fact>)`, i.e. database facts, denote the component entries of the initial state of the story. Based on the story plot and the static schema, the initial state of the narrative can be defined:

```
/* Database Facts – Initial State*/
db(protagonist('Peter')).
db(protagonist('Anne')).
db(supporting('Angel')).
db(supporting('Devil')).
db(student('Peter')).
db(student('Anne')).
db(adviser('Angel')).
db(adviser('Devil')).
db(gender('Peter', male)).
db(gender('Anne', female)).
db(joy('Peter', 5.0)).
db(joy('Anne', 10.0)).
db(place('University')).
db(place('Party')).
db(place('PeterHouse')).
db(place('AnneHouse')).
db(home('Peter', 'PeterHouse')).
db(home('Anne', 'AnneHouse')).
db(current_place('Peter', 'University')).
db(current_place('Anne', 'University')).
db(affection(['Peter', 'Anne'], 0.0)).
db(affection(['Anne', 'Peter'], 0.0)).
```

The state of the world at a given instant consists of all facts about the existing entity instances and their properties (attributes and relationships) holding at that instant. A genre is compatible with an ample choice of (valid) initial states. Different initial states lead to the development of possibly very different narratives, all of which are constrained to remain within the limits of the defined genre.

5.1.1.2. Dynamic Schema

After defining the static schema, the next step in the specification of the story context is the definition of the dynamic schema, which includes the declaration of the basic operations.

A narrative event is defined by a transition from a valid world state S_i to another state S_j, which should also be valid. Changes in the world state must be

limited to what can be accomplished by applying a limited repertoire of operations, which will define what kind of events can occur in the narrative. The operations are formally specified using the STRIPS formalism (Fikes and Nilsson 1971). Each operation is defined in terms of its pre-conditions and post-conditions. Pre-conditions are conjunctions of positive or negative database facts, which must hold at the state in which the operation is to be executed. Post-conditions (effects) consist of two sets of facts: those to be asserted and those to be retracted as a consequence of executing the operation. The events can be either deterministic or nondeterministic. While a deterministic event has only one possible outcome, a nondeterministic event can have multiple outcomes, which are defined by adding alternative sets of effects (post-conditions) to the operations. Guaranteeing the transition between valid states depends on a careful adjustment of the interplay among pre- and post-conditions over the entire repertoire of operations.

The dynamic schema is specified with the following syntax, wherein each operation is defined by an operation frame and an operation declaration:

```
operator_frame(<operator-id>, <operator-name>,
              [<case>:(<entity class or role>;...;
                       <entity class or role>),...,
               <case>:(<entity class or role>;...;
                        entity class or role>]).

operator(<operator-id>,<operator-name>(<parameter list>),
         [<pre-conditions>],
         [<effects>],
         <estimated cost of operation>,
         [<main effects>],[],[]).
```

The operations are specified according to the possible events that can occur on the narrative. In the example given in the previous section, we can easily identify two possible events for the narrative in the first sentence of the plot: "*A young boy named Peter falls in love with a girl named Anne that he met at the university*". Clearly, there is a "meet" event, where a character meets another character, and there is a "fall in love" event, where a character falls in love with another character. We can also think about some pre-conditions that must hold to

these events to occur: in order to meet someone, both characters must be at the same place; and to fall in love, the characters must know and have some affection for each other. Similarly, some post-conditions (effects) can be established: after meeting someone, the character gets to know the other one; and after falling in love, the character begins to love the other one. Both events are deterministic and have only one possible outcome.

Based on these ideas, we can formally specify the operations in a concrete notation:

```
operator(1, meet(CH1,CH2),
        [
           current_place(CH1,PL),
           current_place(CH2,PL),
           not(know([CH1,CH2]))
        ],
        [
           know([CH1,CH2])
        ],
        [],10, [know([CH1,CH2])],[],[]) :-
           db(protagonist(CH1)),
           db(protagonist(CH2)),
           dif(CH1,CH2).

operator(2, fallinlove(CH1,CH2),
        [
           know([CH1,CH2]),
           affection([CH1,CH2],A),
           {A > 5}, {A < 10}
        ],
        [
           not(affection([CH1,CH2],A)),
           affection([CH1,CH2],10)
        ],
        [],10, [affection([CH1,CH2],10)],[],[]) :-
           db(protagonist(CH1)),
           db(protagonist(CH2)),
           dif(CH1,CH2).
```

Apart from the primitive domain operators, an operator can be either an abstraction of one or more specific operators and/or a composition of partially ordered sub-operators. In both cases, they are called complex operators. Abstraction relates a parent operator to one or more child operators. If the parent operator is also a composite operator, its children will inherit its composition. Children operators can also add new operators to the composition, and even include new operators amongst the inherited sub-operators.

Considering the plot example presented in the previous section, we can imagine a possible event where a character attempts and fails to seduce the other one. There are several ways a character can try to seduce someone: a direct and polite approach; a more casual and ironic approach; or even a more aggressive approach. This event can be modeled as a generic operator that has three different specializations (Figure 5.2).

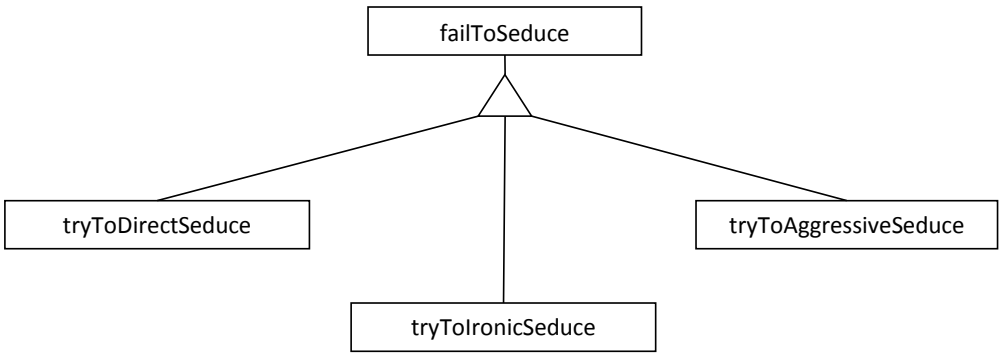

Figure 5.2: Example of generic operator that has three different specializations.

According to these definitions, the specializations of the genetic operator `failToSeduce` can be formally specified in a concrete notation:

```
specialise(tryToDirectSeduce(CH1,CH2),
                             failToSeduce(CH1,CH2)).
specialise(tryToIronicSeduce(CH1,CH2),
                             failToSeduce(CH1,CH2)).
specialise(tryToAggressiveSeduce(CH1,CH2),
                             failToSeduce(CH1, CH2)).
```

In the same plot example, we can also imagine a possible event where a character tries to find more information about another character that he/she just

met by following him/her. This event can be composed of other basic events, which may involve the character going to other places to follow its target. This event can be logically modeled as a composite operator that has four sub-operators (Figure 5.3).

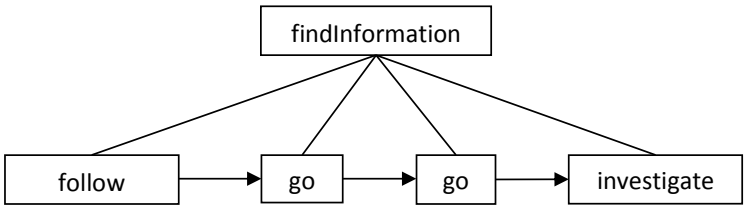

Figure 5.3: Example of composite operator that has four sub-operators.

Based on these specifications, the composite operator for the `findInformation` event can be formally defined:

```
operator(8, findInformation(CH1,CH2),
         [
           not(knowMore([CH1,CH2])),
           affection([CH1,CH2],10)
         ],
         [
           knowMore([CH1,CH2])
         ],
         [],10,[knowMore([CH1,CH2])],
         [
           (f1, follow(CH1, CH2)),
           (f2, go(CH2,PL2)),
           (f3, go(CH1,PL2)),
           (f4, investigate(CH1,PL2))
        ],
        [(f1,f2),(f2,f3),(f3,f4)]) :-
              db(student(CH1)),
              db(student(CH2)),
              dif(CH1,CH2).
```

where the two last parameters of the operator indicate the sub-operators and their partial order. It is important to notice that all sub-operators also must be specified along the dynamic schema, and they can also be composite or abstract operators.

5.1.1.3. Behavioral Schema

The behavioral scheme describes the motives that guide the behavior of the characters in the narrative. It consists of a set of goal-inference rules that capture the goals that motivate the characters' actions when certain situations occur during the narrative. Whenever goals are inferred but not achieved within a plot, a plot composition procedure considers the possible extensions to the plot so that these goals are fulfilled.

An example of goal-inference rule for the narrative idea presented on previous sections could be: "A lonely protagonist wishes to fall in love with another protagonist", which can be formally specified as:

$$\Box(\text{protagonist(CH1)} \wedge \text{protagonist(CH2)} \wedge \text{affection(CH1, CH2)} < 10) \Rightarrow \Diamond(\text{affection(CH1, CH2)} \geq 10)$$

where $\Box$ and $\Diamond$ are the temporal modal operators "always holds" and "eventually holds" respectively; and $\wedge$ is the logical symbol of conjunction.

In the story context, goal-inference rules are specified in a temporal modal logic formalism using a clause of the form `rule(<situation>,<goal>)`, where the `goal` will motivate the agents when a certain `situation` occurs during the narrative. Both `situation` and `goal` are conjunctive lists of literals, denoted using square brackets and "," as separator. The following meta-predicates are used to specify the occurrence of an event or the truth value of a literal at certain times:

`h(T,L)` - the literal `L` is necessarily true at time `T`

`p(T,L)` - the literal `L` is possibly true at time `T`

`e(T,L)` - the literal `L` is established at time `T`

`o(T,E)` - the event `E` occurred at time `T`

Using this notation, the goal-inference rule "A lonely protagonist wishes to fall in love with another protagonist" can be specified in a concrete notation:

```
rule([
       e(i,protagonist(P1)),
```

```
  e(i,affection([P1,P2],A)),
   h({A<10})
],
([T2],
[
  h(T2,affection([P1,P2],10))
],true))
```

Goal-inference rules do not determine specific reactions for the characters – they only indicate goals to be pursued somehow. The events that will eventually achieve the goals are determined by the planning algorithm.

5.1.1.4. Detailed Schema

The final stage of the specification of a story context consists of defining the detailed schema, which includes the definition of the nondeterministic automata used to detail the dramatization of the narrative events.

The events of the narrative are described by means of nondeterministic automata, which are specified and associated with the basic operations defined in the dynamic schema. Each transition in the automaton corresponds to an action that maps a state into a set of possible successor states. The automaton specifies an initial state, at which the event's pre-conditions hold, and a set of final states, at which the event's post-conditions hold. In addition, facts valid at the initial state that are not modified by the post-conditions are also assumed to hold at the final state. The automaton provides therefore various alternatives for the dramatization of each event in a plot, all logically consistent due to the preservation of the correct chaining of pre- and post-conditions.

In each automaton, states are described by invariants, expressed as logical formulae involving dramatization control variables. Transitions between states correspond to basic actions that can be directly performed by the actors. States that have more than one adjacency define local decision points, where users can decide which action the actors should take.

Figure 5.4 shows an example of automaton that describes the dramatization of the event `follow(CH1,CH2)`, where a character CH_1 follows another character CH_2. In the initial state S_1, both characters are in the same place. Then CH_2 goes to

another place and CH_1 stats following CH_2, which leads the automaton to state S_2. Next, CH_2 senses that something is wrong and look back to see what is happening, which induces CH_2 to hide (states S_3 and S_4). If CH_2 noticed that someone was following him/her, he/she enters into the ladies' room (S_7) and CH_1 loses his/her track, which leads the automaton execution to the final state S_8. Otherwise, if CH_2 did not notice CH_1, he/she enters into the classroom (S_5) and CH_1 gets the information about CH_2 classroom (final state S_6). There is also a loop in state $S4$, which makes CH_1 to keep following CH_1 until he/she arrives at the classroom or notices that he/she is being followed and enters into the ladies room.

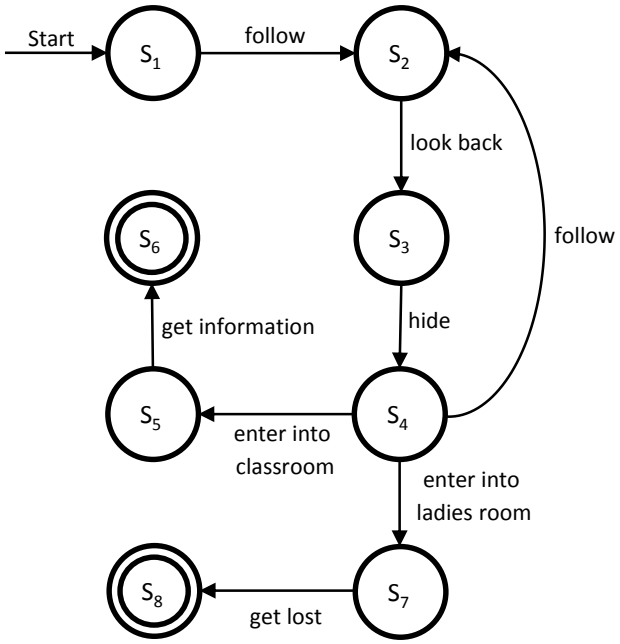

Figure 5.4: Example of automaton describing the dramatization of the event follow(CH_1, CH_2).

The nondeterministic automatons are specified in the GEXF (Graph Exchange XML Format),[4] which is an XML-based language for describing graph structures. The structure of the GEXF file that describes the automatons is illustrated in Figure 5.5.

The specification of the automata demands some authorial effort, but with few states and transitions it is possible to create very flexible dramatizations that preserve coherence within any plot containing the corresponding events.

[4] GEXF File Format - http://gexf.net/format/

```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
  <graph defaultedgetype="directed">
    <attributes class="edge">
      <attribute id="0" title="basicaction" type="string"/>
    </attributes>
    <attributes class="node">
      <attribute id="0" title="state" type="string"/>
    </attributes>
    <nodes>
      <node id="0" label="StateName">
        <attvalues>
          <attvalue for="0" value="StateFacts"/>
        </attvalues>
      </node>
      .
      .
      .
    </nodes>
    <edges>
      <edge id="0" source="0" target="1">
        <attvalues>
          <attvalue for="0" value="BasicAction"/>
        </attvalues>
      </edge>
      .
      .
      .
    </edges>
  </graph>
</gexf>
```

Figure 5.5: Structure of the GEXF file that describes the nondeterministic automatons of the narrative.

5.1.2. Shooting Script Generation

The logical definition of the story context provides the basic information the system needs to automatically generate plots. However, it still needs the video resources to represent them during the dramatization phase. In order to simplify the process of recording the video material required for dramatization, we developed an application that uses the logical definition of the story to automatically generate a shooting script describing how to film the necessary elements (actors and locations) to represent the video-based interactive narrative.

A shooting script is a technical document used in the film industry to specify the sequence of shots that need to be filmed during the production phase

(Swain and Swain 1998). It contains a very elaborate description of all shots, locations, character, action, sound and technical details of the film. A common format for this document is called "Two Column Shooting Script", which consists in dividing the document in two columns: one containing a description of the shot and other with the respective dialog or sound effects. Figure 5.6 shows an example of a two column shooting script used in filmmaking.

Shots	**Shot Description**	**Audio and Dialog**
1	CU of Mother pleading with the father	Mother – sad "*I can't live like this anymore!*"
2	MS of father's reaction	Father – sternly "*We have no options*"
3	LS of father and mother quarreling through a door. Son enters into the frame from the right into foreground in WA and watches them	Mother – upset "*I just can't deal with this violence*" … cries
4	CU of the father	Father – seriously "*What else can we do!*"
5	ECU of the son with an angry look on his face	
6	ECU of the pistol and his fingers on the trigger	

Figure 5.6: Example of a two column shooting script. Abbreviations: CU - close up; MS - medium shot; LS - long shot; WA - wide angle; ECU - extreme close up.

In order to generate a shooting script document, we developed an application that uses the logical definition of the story to simulate all the possible storylines that can be created based on the story context. This process generates a large tree of events, where each node corresponds to an instance of a logical operator representing a story event. The nodes of the tree are then used to instantiate their respective nondeterministic automata (initializing all variables of the basic actions). Then, all basic actions are added to a list structure that contains a logical description of all the possible basic actions that may be performed by the actors during the narrative.

The shooting script document is created based on the list of basic actions. First, the basic actions performed by each actor are grouped and duplicate actions of the same actor with the same or different parameters are removed from the list. Then, for each basic action of each actor a simple natural language sentence is created based on the logical description of the action and a simple text template. For example, the basic action `look(anne, peter)` is converted to the sentence: "*LS of Anne looking at someone*", where the predicate indicates the verb of the sentence and the first variable symbol indicates the character performing the action. The other parameters are generic and may refer to other characters or objects depending on the instance of the action, thus they are omitted and replaced by "someone" or "something". A similar procedure is adopted in dialog actions, with the addition of a step where the content of the dialog is extracted to be incorporated into the dialog column of the document.

Figure 5.7 shows a segment of shooting script automatically generated by the system, where the first column indicates the number of the shot, the second presents a description of the shot, and the third column includes the dialog when necessary. We also added a fourth column that indicates whether the action requires the production of a loop sequence or not. However, the system cannot automatically predict when a loop sequence is required based only in the story context, thus this information needs to be manually added to the document by a human expert.

Shots	**Shot Description**	**Dialog**	**Loop**
1	LS of Anne walking		Yes
2	LS of Anne smiling		No
3	LS of Anne kissing someone		No
4	LS of Anne	Anne – "*I'm waiting for someone*"	No
5	LS of Anne	Anne – "*Well, this is impossible!*"	No
6	LS of Anne	Anne – "*Today was really nice.*"	No

Figure 5.7: Segment of a shooting script automatically generated by the system.

The final shooting script document is divided into sections that separate the shots of each actor individually. In addition, the locations of the narrative that must be filmed are also automatically listed at the end of the document. An

example of a full shooting script document generated by the system can be found in (Lima and Feijó 2014).

5.2. Production

Once the story context has been logically defined and the shooting script has been generated, the next step consists in filming the video resources necessary for the dramatization of the narrative according to the instructions provided by the shooting script document.

The proposed video-based interactive storytelling system is based on the use of video compositing techniques to dynamically create video sequences to represent the story events generated by planning algorithms. In order to compose the scenes in real-time, the system uses as input videos that must pass through a pre-processing phase to remove the background from the videos. The first step of the production phase consists in defining which matting technique will be used in order to extract the visual elements from the background so they can be used in the compositing process.

The matting techniques can be divided in hardware-based (Joshi et al. 2006; Sun et al. 2006; McGuire et al. 2005) and software-based methods (Sun et al. 2004; Gastal and Oliveira 2010; Wang and Cohen 2007). Hardware-based methods usually rely on additional information provided by special equipment, which effectively enhance the efficiency, but increases the production costs. By contrast, software-based methods do not rely on special equipment; they work directly with the visual information provided by the video frames.

Examples of hardware-based matting techniques include the use of an array of cameras (Joshi et al. 2006), which uses the relative parallax between the frames produced by the aligned cameras to capture the foreground objects in front of different parts of the background; another example is the flash matting system (Sun et al. 2006), which uses flash/no-flash image pairs to extract alpha masks based on the observation that only the foreground object has the most noticeable difference between the images if the background is sufficiently distant.

Examples of software-based matting techniques include the Poisson matting algorithm (Sun et al. 2004), which operates directly on the gradient of the matte

based on the assumption that the intensity change in the foreground and background is smooth and thereby the gradient of the matte matches with the gradient of the image; another example is the Shared Matting (Gastal and Oliveira 2010), which collects samples by shooting rays from observed pixels to background and foreground, then the samples that lie along the rays are collected and used for alpha estimation. A comprehensive review on image and video matting techniques is presented by Wang and Cohen (2007).

The proposed method for video-based interactive storytelling receives as input alpha masks, which can be generated by all alpha matting techniques. For the experiments conducted during the development of this thesis, we selected the chroma key matting method, which is the most common matting technique used in the film industry today (Foster 2010). It involves shooting the visual elements in front of a green or blue screen, and then using an algorithm to remove the background from the shot based on the color range of the colored screen.

Once the matting technique has been defined, the next steps of the production phase involve the process of building the film set and filming the raw elements of the narrative (actors' actions and locations) according to the shooting script generated by the system. The next sub-sections describe these steps.

5.2.1. Set Construction and Camera Setup

The construction of the film set involves the process of defining the shooting procedure, building the green screen and placing the cameras in the set.

As previously mentioned, both actors and locations are filmed from different angles in order to give to the system the freedom to dramatize scenes from different angles and apply the basic cinematography concepts during the dramatization of the narrative. In addition, the actors are filmed in front of a green screen, which allows the system to remove the background using the chroma key matting technique and dynamically compose the scenes of the narrative without being restricted by static video sequences. A total of 8 angles of the actors performing their actions must be recorded using a single or multiple cameras in front of a green screen with intervals of 45 degrees (forming a circle around the

subject). The first step of the production phase consists of defining how these 8 angles of subject will be shot.

Four shooting procedures are proposed: (1) full circle filming setup; (2) semicircle filming setup; (3) one-quarter filming setup; and (4) single camera filming setup.

The full circle filming setup consists of using 8 cameras to record the action simultaneously (Figure 5.8). This can be done by building a cylindrical structure with the interior coated with a green screen fabric. The cameras are placed outside and around the structure with the lens embedded in small holes built in the cylindrical wall with intervals of 45 degrees. The actor is placed at the center of the structure and can perform the actions while the cameras record his performance. The cameras have to be placed outside to avoid that two facing cameras film each other. Lights, microphones and other filming equipment can be positioned over the structure.

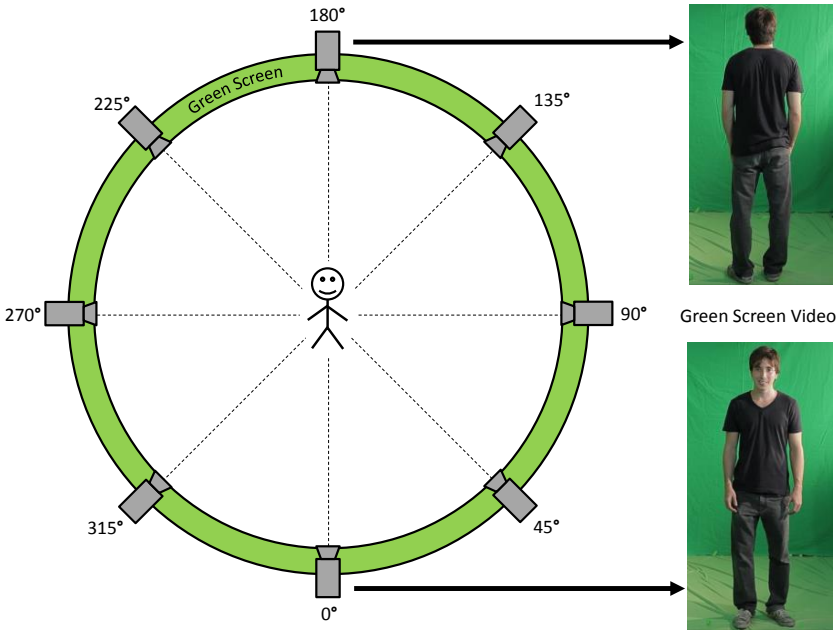

Figure 5.8: Full circle filming setup.

The main advantage of the full circle filming setup is that it makes easier and simplifies the work for both cinematographers and actors. The actors only need to perform the actions once and the cameras record all angles simultaneously without requiring adjustments in the camera setup. The main drawback is the growth of the productions costs. It requires eight equivalent cameras and the construction of the cylindrical film set.

The second shooting procedure is the semicircle filming setup (Figure 5.9). It consists of a simplified version of full circle filming setup, where only half of the circle is created. In this method, three green screen walls are built and arranged as illustrated in Figure 5.9. Inside of the walls, 5 cameras are placed forming a semicircle around the subject. The two cameras placed at straight line angles (0° and 180°) are positioned in two small holes created in the parallel walls. The actor is placed at the center of the semicircle and five angles of the actions are recorded simultaneously. In order to obtain the missing angles, there are two options: (1) the missing angles are obtained by flipping the videos of cameras 45°, 90° and 135° horizontally; or (2) the actor turns 180 degrees and repeats the action.

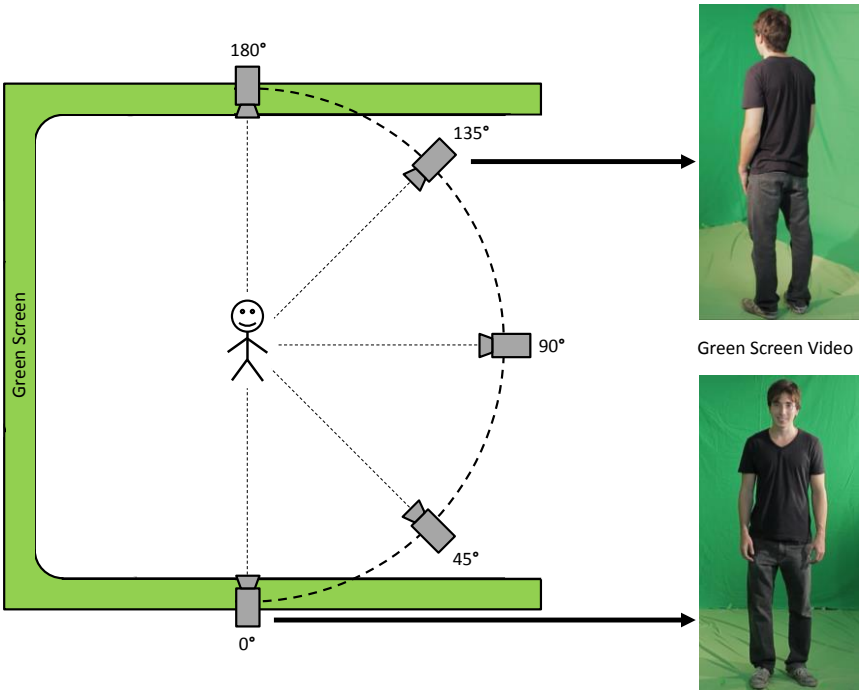

Figure 5.9: Semicircle filming setup.

The main advantage of the semicircle filming setup is the reduction of the production costs. It requires less cameras than the full circle setup and can produce similar results without increasing the work of actors and cinematographers. The main disadvantage is that the trick of flipping the videos of some cameras to produce the missing angles only work well when the actors, their clothes and props are fully symmetrical; otherwise it may break the continuity of the film. For example, if an actor is holding a handbag in his right hand when filmed from the angle of 45 degrees, in the flipped video of 315 degrees he will

appear holding the handbag in his left hand, which may confuse the audience. In order to avoid this problem, when the actors are not symmetrical, the actor has to repeat the action two times to record his performance from all camera angles.

The semicircle setup can also be used in the cylindrical structure of the full circle setup, and indeed it may produce better results regarding the quality of the green background. However, building three green screen walls may be easier that constructing the cylindrical structure.

The third proposed shooting method is the one-quarter filming setup (Figure 5.10). It consists of a more simplified version of semicircle filming setup, where only one-quarter of the circle of cameras is created. In this method, only two green screen walls are built and positioned as illustrated in Figure 5.10. Next to the walls, 3 cameras are placed forming one-quarter of a circle around the subject. In this camera setup, three angles of the actions performed by the actors are recorded simultaneously. In order to obtain the missing angles, the actor has to repeat the actions two or four times. If the actor and his clothes/props are fully symmetrical, the action is repeated two times to obtain the angles of a semicircle, and the angles of the other side of the semicircle are obtained by flipping the recorded angles. Otherwise, if the actor is not symmetrical, the action has to be repeated four times so that all angles of the actor are recorded.

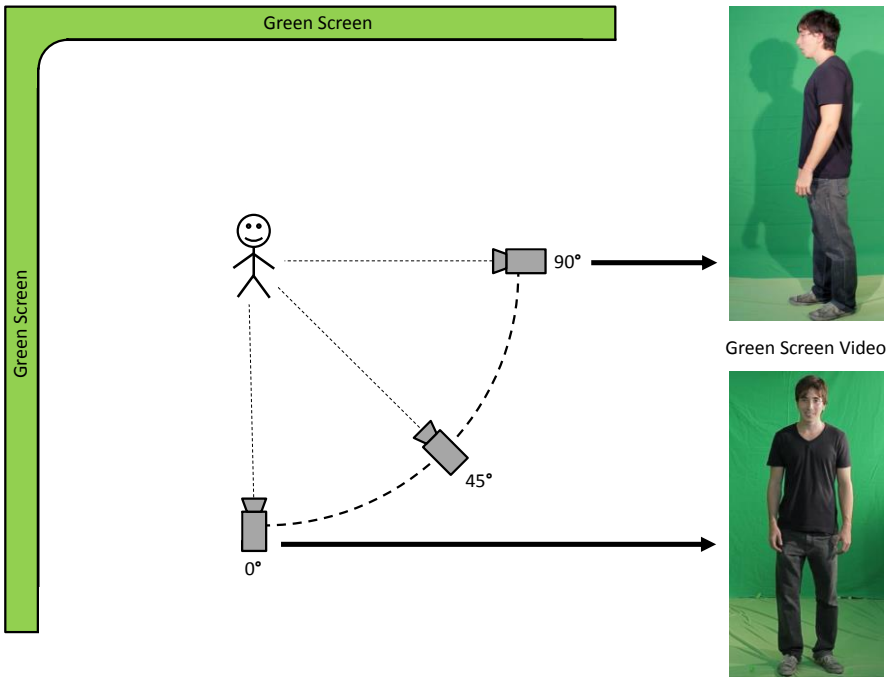

Figure 5.10: One-quarter filming setup.

The main advantage of the one-quarter filming method is the notable reduction of the production costs. However, it significantly increases the work of actors, who need to repeat the actions several times. Although there is no need to the actions to be repeated exactly the same way, it is important that they have some degree of similarity, which may be difficult to be achieved by nonprofessional actors.

The last shooting method is the single camera film setup (Figure 5.11), where all the angles of the action are recorded using only one camera. The method uses a single green screen wall and the actor is positioned between the wall and the camera (Figure 5.11). In order to record all required angles, the actor has to perform the action, turn 45 degrees, repeat the action, and do this until he completes the eight angles of the circle. Again, the number of times the actors has to repeat the actions can be reduced in half by flipping the recorded angles, but only if the actor and his clothes/props are fully symmetrical.

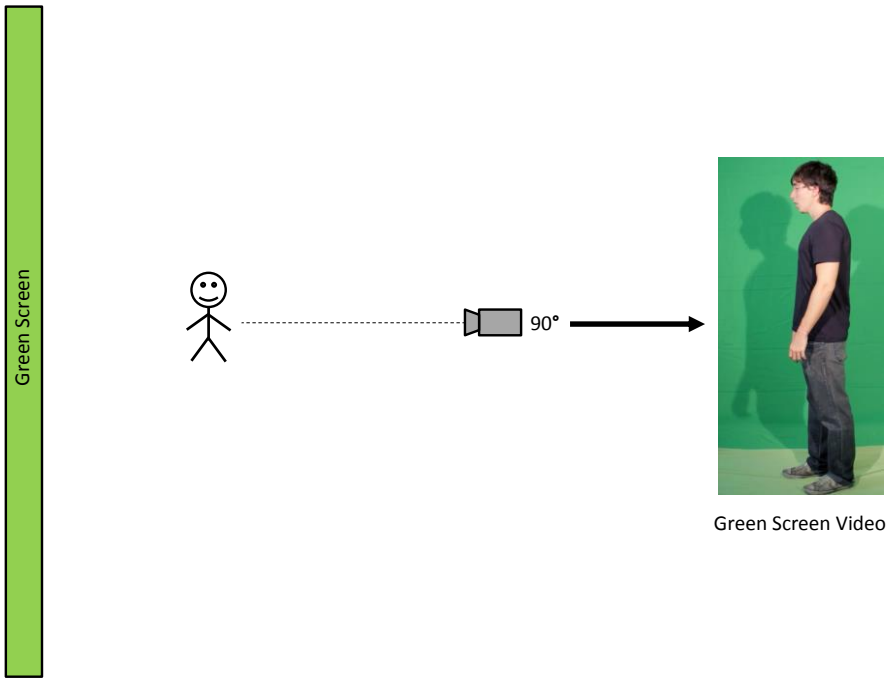

Figure 5.11: Single camera film setup.

The great advantage of the single camera setup is the huge reduction of the production costs. However, it excessively increases the number of times the actors have to repeat the actions, which is a very exhaustive task and may compromise

their performance. Although it is possible to film the videos of a full interactive narrative using this method, it is only recommended for testing purposes.

In the four shooting procedures, the distance of the cameras to the subject depends on the focal length of the camera. A longer focal length produces a narrower angle of view, whereas a shorter focal length produces a wider angle of view. All the cameras must be positioned in the film set in a way to capture the whole body of the actors in the frame.

5.2.2. Shooting Actions

Once the filming method is defined and the film set is constructed, the next step of the production phase consists of shooting the actors performing their actions. The shooting script generated by the system provides a basic description of the actions that have to be recorded, however it is a task of the film director to supervise and give more detailed instructions to the actors about the actions they need to perform.

If a full circle filming method is used, the actors only have to repeat each action once; otherwise, they have to repeat the same action more times so that all required angles are recorded. When the action needs to be repeated, there is no need to the action to be performed exactly the same as in the previous shot. The dramatization system will not switch between different angles during the exhibition of an action. The camera angle will be selected only at the beginning of the action, so the audience will not notice differences in the actors' performance. However, it is important that the performance have some degree of similarity in the different angles to avoid inconsistencies when transiting between different actions. The film director must supervise the shooting process to guarantee the coherence in the recorded actions.

One of the main rules that must be obeyed when shooting the actors is that all actions must be performed in-place, which means that the actor must stay always on the same position while acting. During the dramatization, the system will be in charge of automatically positioning and controlling the movement of the actors, so it is important that all actions be recorded without significant changes in the position of the actors. For most part of the actions this is not a problem,

however, some actions naturally require movements of the actor on the stage. A very common example is the basic action of walking, which is present in most part of the narratives and requires the explicit movement of the actors. In order to record this type of action, we propose the use of a treadmill positioned at the center of the filming set (Figure 5.12). The treadmill allows the actors to walk or run while staying in the same place. In our experiments, we used a simple exercises treadmill, which produced good results and allowed us to film the actors walking and running in-place. However, for more professional results it is recommended the use of a more personalized treadmill that could be embedded in the structure of the film set. In addition, the color of the treadmill must be the same as the color of the background, so it can be easily removed from the video during the post-production phase.

Figure 5.12: In-place walking action being performed over a treadmill.

Another important aspect to consider when filming the actors is whether the videos of the actions need to be in loop or not. A looping video consist of sequence frames that can be repeated continuously without jumps. Simple and direct actions, such as talking, looking and grabbing, do not require looping videos, because the end of the video indicates the end of the action. However, more dynamic actions, where the end of the video not necessarily indicates the end of the action, require videos ready to be played in loop by the system. For

example, it is not possible to predict the precisely length for the videos representing a walk action because it depends on the distance traveled by the virtual actor, which may vary from scene to scene. Another example is the idle action, which is used to represent supporting characters that are participating of the scene without performing any specific action. Again, the length of the video depends on the length of the scene. In these cases, the videos must be prepared to be played in loop, so the system can represent the actor walking any distance or staying in the idle position for an indefinite time.

Although the actual process of creating the looping videos is done in the post-production phase, it is important that the recorded material allows the creation of looping sequences. The recommendation is to film long sequences of the actions that require looping sequences being repeated by the actors, so the editors will have sufficient material to work with during the post-production phase. In our experiments, the looping actions were recorded for 5 minutes, which provided enough material to create the looping videos.

5.2.3. Shooting Locations

The locations are composed of a set of video or image layers representing the environments where the story events can occur. Usually, outdoor locations are represented through videos, which are able to reflect the natural dynamism of the environment (e.g. leaves being moved by the wind, people walking in the distance, birds flying around), and indoor locations that do not include dynamic elements are represented by static pictures. The process of shooting the locations of the narrative is easier than shooting the actors. It does not require the construction of a filming set and the actors do not have to go with the filming crew to the physical location.

The shooting script generated by the system provides a basic description of the locations that must be filmed based on the logical context of the story. Similarly to the process of filming the actors, each location of the narrative also must be recorded from eight angles with intervals of 45°, forming a circle around the stage (Figure 5.13). The radius of the circle depends on the focal length of the camera. The longer the focal length of the cameras, the larger the radius of the

circle must be to cover the entire stage on the frame of all the cameras. In cases of small locations (e.g. corridors, small rooms) or to reduce the production work, some angles can be omitted. In these cases the dramatization system will automatically avoid showing the scenes from the missing angles.

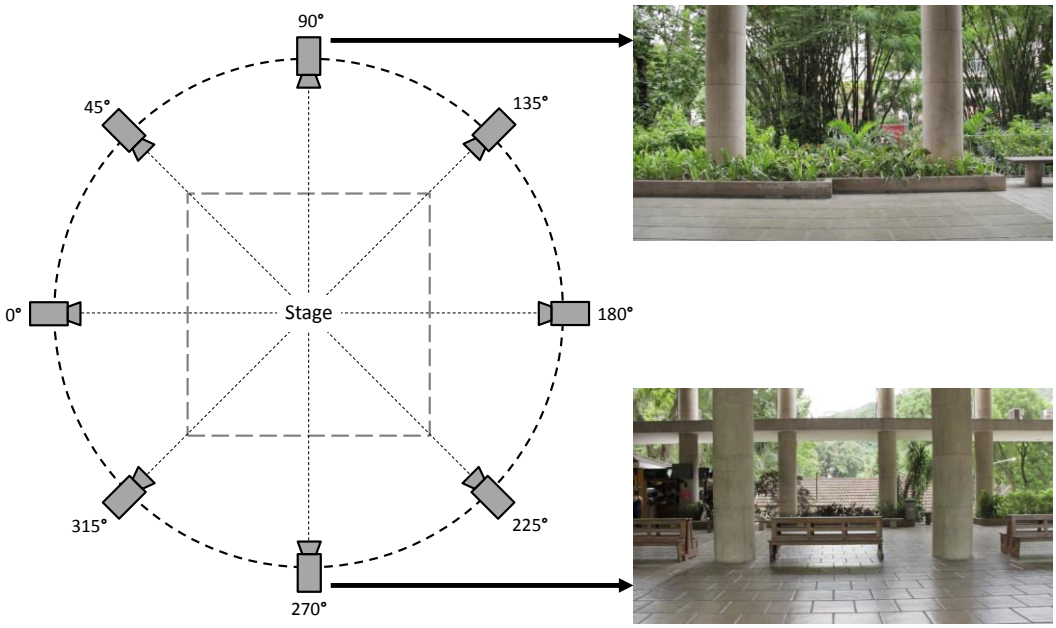

Figure 5.13: Camera placement for filming locations.

The locations can comprise one or more layers depending on the existence of objects that belong to the location and must overlap the characters acting in the scene (e.g. tables, pillars, doors). Although the layers are defined and created during the post-production phase, the filming crew must always imagine the objects that would be part of additional layers in order to film the location appropriately. Figure 5.14 shows an example of location composed of two layers (L_1 and L_2), where L_2 represents a table that overlaps the background L_1 and the characters acting in the scene.

Although it is possible to create layers with dynamic moving objects using videos, it is recommended to use photos with only static objects in order to simplify the work in the post-production phase. The main problem is the difficulty of separating the foreground elements from the background. While shooting the actors, the green screen was used to simplify this task. However, in this case there is no green screen so the editors have to separate the foreground layers manually frame by frame. An alternative to film locations where dynamic foreground layers

are really necessary is the placement of a green screen in the back of the foreground elements. In this case, two videos must be recorded, one without the green screen and other with the green background, keeping exactly the same camera position and angle in both videos.

Figure 5.14: Location with 2 layers (L_1 and L_2).

5.2.4. Shooting Static Scenes

Scenes that include complex interactions between characters or other objects that may not be adequately composed by the system in real-time can be represented as static scenes. This type of scene consists of a prerecorded video of the entire scene, including characters, film set, camera movements and different shots of the actions as a traditional film. When an event that is represented by a static scene is generated by the planning algorithms, the dramatization system will exhibit the prerecorded video instead of compositing the event in real-time.

The process of shooting static scenes is based on the master scene filming method. This method involves the filming of the entire scene with the master shot (a shot that includes the whole setting) along with coverage (shots that reveal different aspects of the action and use only view angles that are different from the master shot). In this way, the dramatization system always has two shots of each scene. If the agent detects a problem in the coverage shots, a new shot of the same action can be extracted from the master scene (Figure 5.15). The main motivation under the use of the master scene method is to make sure that there are no mismatches of continuity and no gaps between the actions. Furthermore, this

method gives to the editor the freedom to creatively cut and alter the pacing, the emphasis, and even the point of view of the scenes. Filming a scene with the master scene method can be done using a single camera or multiple cameras (Mascelli 1965). According to Brown (2011), the master scene method is used in probably 95% of narrative films shots today.

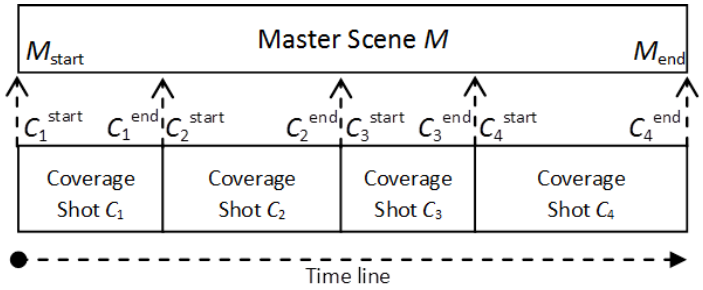

Figure 5.15: The master scene structure.

The choice of which scenes will be represented using static scenes is a decision of the film director. The proposed video-based dramatization system supports both dynamic composed scenes and static video sequences. An interactive film may be entirely represented using static scenes, however, as previously mentioned, it may face problems related to the lack of interactivity, story diversity and high production costs.

5.3. Post-Production

The post-production phase involves the process of editing the raw material captured during the production phase, including the process of removing the green screen background of the videos, separating the actors' actions and locations into individual video files, defining the structure of the virtual locations and associating the video files with the corresponding actors and locations. The next sub-sections describe these steps.

5.3.1. Editing Actions

The first step of the post-production phase comprises the process of editing the videos of the actors' actions filmed during the production phase. It involves

the task of removing the background of the videos to create alpha masks, the process of creating looping sequences when required, and the task of exporting the videos to be used by the system.

As a traditional video production, the first step of the editing process consists in extracting the content from the raw material, removing unnecessary parts and selecting the best shots when more than one shot of the same action was taken. This process can be done using any video editing software (e.g. Adobe Premiere, VirtualDub, Final Cut). In this step is also important to organize the video material, separating the videos of each actor and action in separated files and folders.

Once the material has been edited and organized, the next step of the process consists in removing the background of the videos to create alpha masks. The alpha mask is a video that encodes the clipping region that separates the actor from the background in the original video. Each frame of the alpha mask is a grey scale image in which black represents fully transparent pixels, white represents fully opaque pixels, and grey pixels represent a corresponding level of opacity. Figure 5.16 shows an example of alpha mask generated for its corresponding video source.

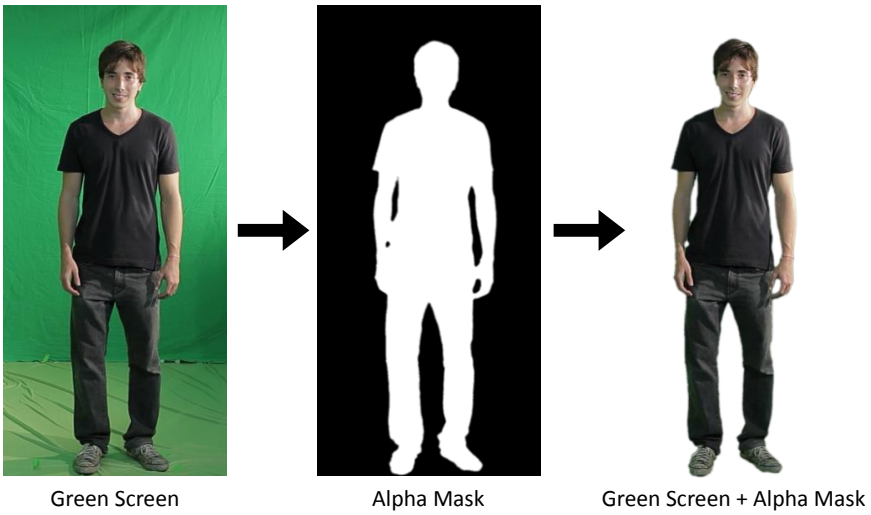

Figure 5.16: Example of alpha mask extracted from the green screen video.

The alpha masks are created using the chroma key matting technique. The most common video editing applications, like Adobe After Effects, Final Cut Pro and Pinnacle Studio include some tools that uses chroma key algorithms to

generate alpha masks. When the foreground object is quite solid with simple and sharp boundaries and is fully opaque, alpha mask can be easily extracted by the algorithms based on the estimation of the background color distribution. However, in some cases, objects may have intricate boundaries, such as hair strands and fluffy toys, or have semi-transparent parts, such as glasses and transparent clothes. In these cases, the extracted object may suffer from "color-spill", or some parts along the boundaries of the object might be cut out. In this way, manual adjustments are required in some situations to improve the results produced by the chroma key algorithms. In our experiments, the Adobe After Effects was used to create the alpha masks (Figure 5.17).

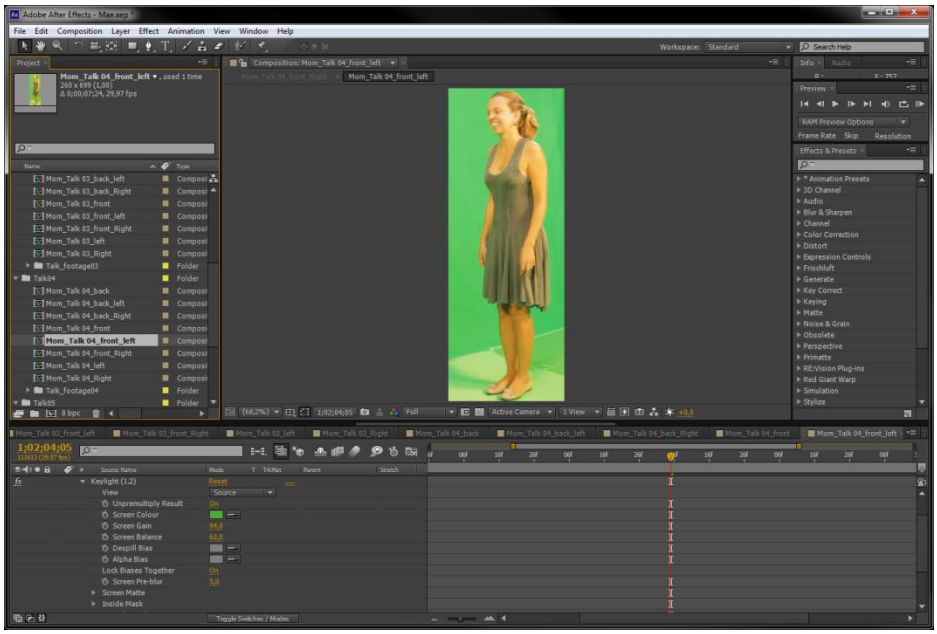

Figure 5.17: Adobe After Effects user interface.

Once the alpha masks have been created, the next step of the post-production phase involves the creation of looping sequences for the actions that may be played in loop during the dramatization of the narrative. This task could be manually done by observing and searching for similar video frames to march and connect the last frame of the video with the first frame. However, it would be a very time consuming task and, as it is not common task on the film industry, there is no available tools to assist the human editor. In order to simplify this process, an application was developed to automatically detect looping sequences (Figure 5.18).

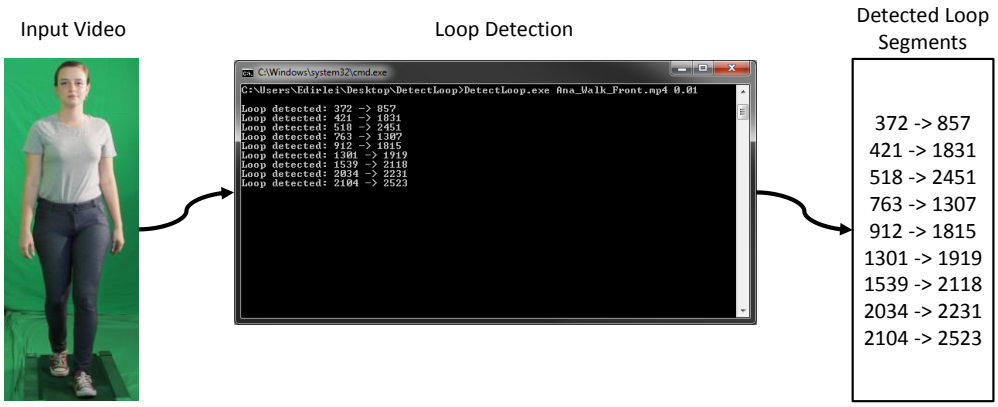

Figure 5.18: The loop detector tool.

The loop detector application receives as input the video segment to be processed and a similarity error factor α. The algorithm searches for loop segments by comparing all the frames of the video to determine the degree of similarity between them using the correlation coefficient metric:

$$F_{coor}(F_x, F_y) = \frac{\sum_i (F_x^h(i) - \overline{F_x^h})(F_y^h(i) - \overline{F_y^h})}{\sqrt{\sum_i (F_x^h(i) - \overline{F_x^h})^2 \sum_i (F_y^h(i) - \overline{F_y^h})^2}}$$

where $F_x^h(i)$ and $F_y^h(i)$ are the histogram values in the discrete interval i and $\overline{F_x^h}$ and $\overline{F_y^h}$ are the histogram bin averages. High values of correlation represent a good match between the frames, that is: $F_{coor}(F_x, F_y) = 1$ represent a perfect match and $F_{coor}(F_x, F_y) = -1$ a maximal mismatch. A frame interval is considered a loop segment only if $F_{coor}(F_x, F_y) > 1 - \alpha$.

The output of the loop detector application is a list of frame intervals of all loop segments detected in the input video. The human editor can then use this list to compare the segments and select the best loop sequence. Usually, the smaller sequences are the best options.

Once the alpha masks and looping sequences have been created, the final step is the process of exporting the videos. Only the frame region occupied by the actors in the videos must be exported. This is very important because the dramatization system uses the height of the video to estimate the real height of the actor (in pixels). The alpha masks and the videos must be exported to separated files and both must be encoded using the H.264/MPEG-4 Part 10 video compression format (ITU-T 2013), which is currently one of the most commonly used formats for the recording, compression, and distribution of video content.

Figure 5.19 illustrates the results of the action editing phase. For each video of an action filmed from a specific angle, two new video files are created: one containing the clipped region of the actor and the other with the corresponding alpha mask that will be used in the compositing process.

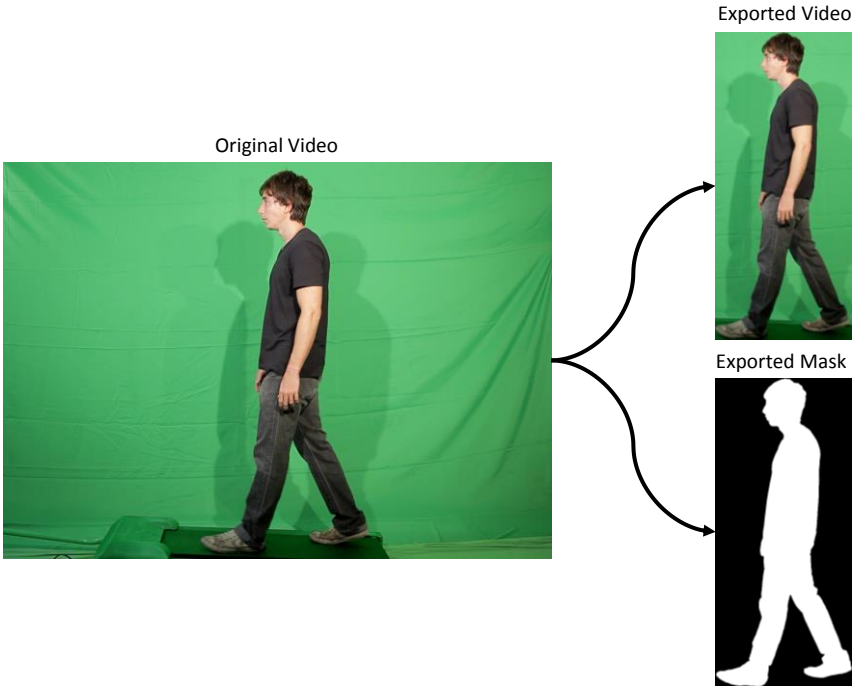

Figure 5.19: Example of results of the action editing phase.

Once all videos have been exported, they must be associated with their respective actors and actions. This information is defined in as XML file, which is presented in Section 5.3.3.

5.3.2. Editing Locations

The second step of the post-production phase involves the process of editing the videos or image layers of the locations where the events of the narrative can happen and creating the basic geometrical structure of the environment by defining the waypoints of each location.

Layers are only necessary when the location contains objects that belong to the environment and must overlap the characters acting in the scene (e.g. tables, pillars, doors). The task of creating layers is similar to the process of separating the characters from the background, but usually in this case there is no green screen background and the objects must be manually separated. However,

generally layers are only required in interior locations or they are composed of only static objects, which allows the layer to be composed of a single frame, simplifying the editing task. Once the layer elements have been separated from the background, an alpha mask must be created to define the clipping region of the layer. Similarly to the actors' actions, each location layer is composed of a video or image representing the visual elements of the layer and its respective alpha mask defining the clipping region (Figure 5.20). Locations composed of a single layer do not require alpha masks.

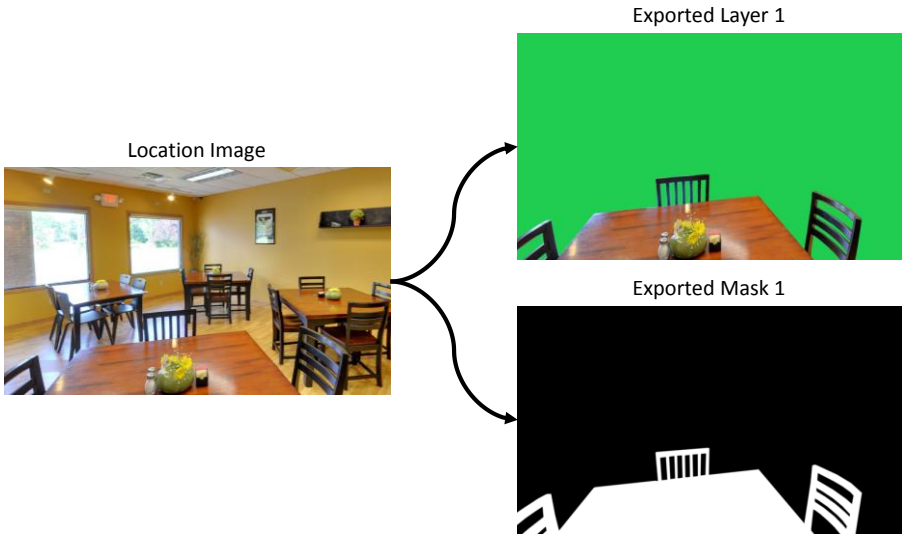

Figure 5.20: Location with a foreground layer defined by an image and its respective alpha mask.

Once the location layers have been defined, the next step consists in defining waypoints in the locations. The waypoints are used to define the basic geometrical structure of the locations, indicating where characters can be placed during the compositing process. This strategy is similar to the one used in games for navigation purposes, which simplifies the execution of path finding algorithms (Millington and Funge 2009).

In the proposed system, each location has a set of waypoints structured in the form of an undirected graph, where the vertices represent the waypoints and the edges represent the connections between the waypoints. There are three types of waypoints: (1) entrance/exit waypoints, which are used as a point of entrance/exit to characters entering/leaving the scene; (2) acting waypoints, which

are used as a point of reference to place characters that are performing some actions in the scene; and (3) connection waypoints, which are used to create paths between the other waypoints. Figure 5.21 shows an example of location that contains a graph connecting five waypoints (W_1, W_2, W_3, W_4 and W_5), of which W_4 and W_5 are entrance waypoints, and W_1, W_2 and W_3 are acting waypoints.

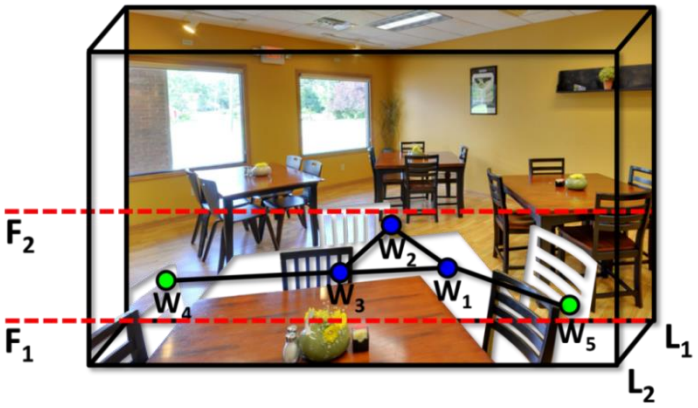

Figure 5.21: Location with 5 waypoints (W_1, W_2, W_3, W_4 and W_5). The front line F_1 and the far line F_2 delimit the region for waypoints.

Each waypoint gives information about its specific position in the location and the orientation that a character occupying that position must assume. The locations also contains the definition of a front line and a far line (F_1 and F_2 on Figure 5.21), which delimits the region where characters can be placed during the compositing process. Both far and front lines include the definition of the relative size that characters must have when they are placed over the lines. This information is used by compositing algorithm to estimate the size that characters must have when they are placed at any position of the scene.

The process of manually defining and annotating the positions of all waypoints may be a very time consuming task considering that all waypoints must be properly placed in the eight camera angles of each location. In order to simplify this process, we developed an interactive tool for waypoint placement (Figure 5.22). The application allows users to place and adjust waypoints in the eight camera angles simultaneously by displaying the location in a set of windows that reflect the circular structure of the available angles of the location. The tool also allows users to define the type, angle and connections between the waypoints. In

addition, the software also includes some facilities to assist users in defining the front and far lines for the locations.

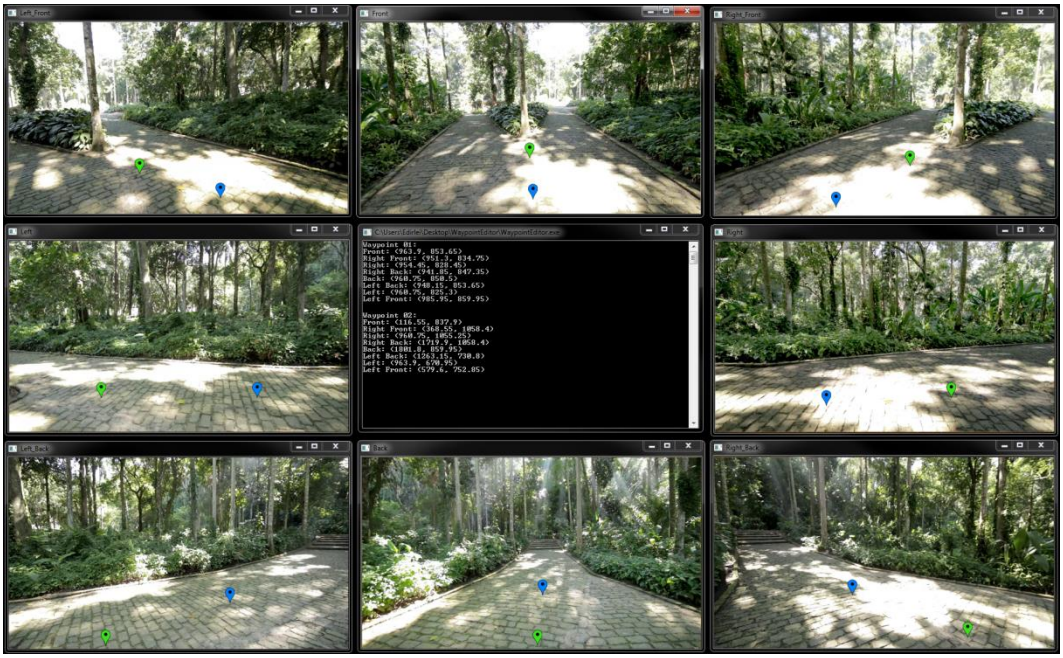

Figure 5.22: The interactive tool for waypoint placement.

Once the layers and waypoints have been established, they must be associated with their respective locations. This information is defined in an XML file, which is presented in Section 5.3.4.

5.3.3. Actor Definition

Actors are defined in an XML file, which associates the video files of actions with the logical definition of an actor entity. The XML file is composed of three main elements:

- **Actor**: represents an actor entity, which is identified by its attribute `name`. Each `Actor` contains a `Location`, defining the initial location of the actor, and a list of `Behaviors` representing the actions the actor can perform;
- **Behavior**: represents an action that an actor can perform and is identified by its attribute `name`. Each `Behavior` contains a set of videos representing the action from different camera angles;

- **Video**: represents a video of an action filmed from a specific camera angle identified by its attribute `type`. Each `Video` contains a `VideoFile` indicating the video of the action and a `MaskFile` indicating the respective alpha mask of the action.

The structure of the XML file that describes the actors of the narrative is illustrated in Figure 5.23. Each character of the story is defined by an `Actor` node, which is identified by the name of the character and contains a definition of the initial location of the character (`Location`) and the actions that can be performed by the actor (`Behaviors`). Each behavior is defined by a name and it is composed of a set of `Video` nodes that represent the action from different angles. Each video is identified by the name of the angle and contains two child nodes that indicate the path to video file of the behavior (`VideoFile`) and the respective path to the alpha mask video file (`MaskFile`). The behaviors can also include some additional parameters, such as the speed of a walk behavior or external audio files to be executed while the behavior is being performed by the actor.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<ActorDatabase name="Video-Based Interactive Narrative">
  <Actor name="ActorName">
    <Location>ActorLocation</Location>
    <Behaviors>
      <Behavior name="BehaviorName">
        <Video type="AngleName">
          <VideoFile>Data\Behavior_Video.mp4</VideoFile>
          <MaskFile>Data\Behavior_Video_Mask.mp4</MaskFile>
        </Video>
        .
        .
        .
        <Parameter1>0.2</Parameter1>
        .
        .
        .
      </Behavior>
      .
      .
      .
    </Behaviors>
  </Actor>
  .
  .
  .
</ActorDatabase>
```

Figure 5.23: Structure of the XML file that describes the actors of the narrative.

An example of an XML file defining the characters of an interactive narrative can be found in (Lima and Feijó 2014).

5.3.4. Location Definition

Locations are also defined in an XML file, which associates the image or video layers of locations with the logical definition of a location. The XML file is composed of three main elements:

- **Location**: represents a location, which is identified by its attribute `name`. Each `Location` contains a set of videos or image layers representing the location from different camera angles;
- **Video**: represents the location filmed from a specific camera angle identified by its attribute `type`. Each video contains a list of `Layers` representing the layers of the location, a list of `Waypoints` indicating the available waypoints, and a definition of a `HorizonLine` and a `FrontLine`, which indicate the far and front lines of the location;
- **Layer**: represents an image or video layer of the location ordered according to its `zIndex`. Each `Layer` contains a `LayerFile` indicating the image or video of the layer and a `LayerMask` indicating the respective alpha mask of the layer;
- **Waypoint**: represents the structure of a waypoint identified by its attribute `name`. Each `Waypoint` is defined by a `type`, position (`x` and `y`), `angle`, `zIndex` and contains a set of `Connections` indicating connected waypoints.

The structure of the XML file that describes the locations of the narrative is illustrated in Figure 5.24. Each location of the story is defined by a `Location` node, which is identified by the name of the location and contains a set of `Video` nodes that represent the physical location from different angles. Each `Video` is identified by the name of the angle and contains a set of `Layer` nodes representing the image or video layers that compose the location. Each `Layer` has two child nodes that indicate the path to video or image file of the layer (`LayerFile`) and the respective path to the alpha mask (`LayerMask`). The `Video` nodes also contain the

definition of the far and front lines delimiting the region where characters can act during the dramatization (`HorizonLine` and `FrontLine`), and a set of `Waypoints` indicating the exact positions where they can be placed. Each `Waypoint` is defined by a `name`, a `type`, its position (`x` and `y`), the `angle` that characters must assume when occupying its position, and an index of its rendering order (`zIndex`). Each `Waypoint` also contains a set of child nodes indicating the waypoints that are connected to the current waypoint (`Connection`).

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<LocationDatabase name="Video-Based Storytelling">
  <Location name="LocationName">
    <Video type="AngleName">
      <Layers>
        <Layer zIndex="0">
          <LayerFile>Data\Layer_Video.mp4</LayerFile>
          <LayerMask>Data\Layer_Video_Mask.mp4</LayerMask>
        </Layer>
        .
        .
        .
      </Layers>
      <HorizonLine position="0" size="0"/>
      <FrontLine position="0" size="0"/>
      <Waypoints>
        <Waypoint name="WaypointName" type="WaypointType" x="0"
                                      y="0" zIndex="0" angle="0">
          <Connection>ConnectedWaypointName</Connection>
          .
          .
          .
        </Waypoint>
        .
        .
        .
      </Waypoints>
    </Video>
    .
    .
    .
  </Location>
  .
  .
  .
</LocationDatabase>
```

Figure 5.24: Structure of the XML file that describes the locations of the narrative.

An example of an XML file defining the locations of an interactive narrative can be found in (Lima and Feijó 2014).

5.3.5.
Static Scenes Definition

Static videos that represent prerecorded scenes ready for presentation are also defined in an XML file, which associates the static coverage video of the action and master scene video with its respective logical story event.

The structure of the XML file that describes the static scenes of the narrative is illustrated in Figure 5.25. Each static scene is defined by a `Scene` node, which is identified by the logical sentence that describes the story event (`event`) and a boolean property identifying loop scenes (`loop`). Each `Scene` has two child nodes that indicate the path to video file of the coverage video (`VideoFile`) and the respective path to the master scene video (`MasterSceneFile`).

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<StaticSceneDatabase name="Video-Based Storytelling">
  <Scene event="EventDescription" loop="false">
    <VideoFile>Data\Static_Video.mp4"</VideoFile>
    <MasterSceneFile>Data\Static_Master.mp4"</MasterSceneFile>
  </Scene>
  .
  .
  .
</StaticSceneDatabase>
```

Figure 5.25: Structure of the XML file that describes the static scenes of the narrative.

An example of an XML file defining the static scenes of an interactive narrative can be found in (Lima and Feijó 2014).

5.3.6.
Narrative Resource Pack

Video-based interactive narratives are composed of several video files, which are indexed and logically associated with characters, locations and events of the story through XML configuration files. Both video and XML files are stored in a narrative resource pack, which consists of a single compressed file that contains all the resources used by the dramatization system to represent the interactive narrative.

In order to create a narrative resource pack, all the resources of the narrative must be compressed using standard .ZIP file format. This file must contain the XML files identifying actors, locations and static scenes of the narrative. These files must be named according to the following conventions:

- `Actors.xml` – XML file that contains the description of the actors of the narrative;
- `Locations.xml` - XML file with the definition of the locations of the narrative;
- `Static.xml` – XML file that contains the description of the static scenes of the narrative.

Video files can be organized in any structure of directories inside of the resource pack as long as their file paths were properly defined in the XML files.

The narrative resource pack file is placed at the story server and it is automatically accessed by story dramatization clients. If a client does not have the resource pack of the requested narrative, the pack will be automatically downloaded and extracted by the client and will be used for the dramatization of the narrative.

5.4. Conclusion

This chapter presented the proposed process for the production of video-based interactive narratives, describing how to write and film an interactive story. In addition, some computational tools to assist the author during this process were described.

The specification of the story context undoubtedly requires some knowledge of programming and planning, which would limit the process of writing new stories to programmers. However, we believe that story specification should be a cooperative work involving both programmers and traditional story writers. While the author should be responsible for the creative process of having ideas for the story, the programmers should be responsible for codifying these ideas as a logical story context. In addition, we believe that the cooperation between story writers, programmers and story generation algorithms can positively affect the creative

process of the author by instigating a co-creation process, where the authors may embrace the output of the story generator algorithms as a contribution to the space of possible stories, changing their initial authorial intent and accepting it as a fundamental part in shaping and constraining the story space.

The production and post-production phases involve several filmmaking professionals, such as producers, actors, directors, camera operators and editors, which are in charge of constructing the film set, acting, shooting and editing the necessary actions, locations, and static scenes. Undoubtedly, these tasks require a considerable amount of work, especially for the people involved in the post-production phase, where all videos have to be edited and prepared to be used during the dramatization. Actors also have to adapt themselves to the production process, which require them to always act alone in front of a green screen performing generic actions without a predefined context, which differs from the way they are used to performing in traditional film productions.