

Please cite as:

Ranon R., Chittaro L., Buttussi F. Automatic Camera Control meets Emergency Simulations: an Application to Aviation Safety, Computers and Graphics, Vol. 48, May 2015, pp. 23–34.

Automatic Camera Control meets Emergency Simulations: an Application to Aviation Safety

Abstract

Computer-based simulations of emergencies increasingly adopt 3D graphics to visualize results and thus generate complex dynamic 3D scenes with many potentially parallel events that affect large groups of virtual characters. To understand the portrayed scenario, a viewer could interactively control a flying camera or switch among a set of virtual cameras that have been previously placed at modeling time. The first solution imposes a cognitive load on the viewer that can distract him/her from the analysis task, and (s)he might miss events while moving the camera. The second solution requires additional work in the modeling phase, and even a very large number of cameras could fail to correctly frame events because of dynamic occlusions. More sophisticated automatic camera control methods could help, but the methods in the literature are designed for sequential dialogue-like events that involve at most two or three characters and therefore would not work. In this paper, we present a fully automated, real-time system that is able to monitor events in emergency simulations, select relevant events based on user-provided filtering rules, and control a virtual camera such that the events of interest are properly presented to the viewer. To illustrate how the system works in practice, we also describe the first application of automatic camera control to the domain of aviation safety.

Keywords: automatic camera control, emergency simulations, aviation safety

1. Introduction

Computer-based simulations of emergencies are increasingly used for a variety of purposes, including planning, prediction of outcomes, accident investigation, and training. Systems have begun to adopt realistic 3D graphics to visualize simulation results (e.g., [1, 2, 3]), thereby generating complex, dynamic 3D scenes with many potentially parallel events affecting large groups of virtual characters. Presenting the resulting animations to a viewer in an effective manner is thus challenging.

The traditional approach to the visualization of 3D simulations is to place multiple virtual cameras in the scene at modeling time and switch among them at run time to observe the different events that occur. However, as the spatial complexity of the scenario and the number of events increase, even a very large number of cameras could fail to correctly frame many events, e.g., because of dynamic occluders that prevent any of the pre-defined cameras from adequately capturing some of the action. Moreover, manually placing the virtual cameras can require a considerable modeling effort that in general must be repeated for each simulation. Even for the same simulation, multiple camera setups may be necessary based on what features a viewer finds the most interesting. For example, a safety expert could be interested in how the entire emergency egress of a crowd from a building evolves, while a firefighter who uses the same simulation for training would need to focus on details that are relevant to first response duties in the field such as the location and evolution of fires.

An alternative solution is to let the viewer interactively control a flying camera during the simulation. However, this approach imposes a cognitive load on the viewer that can distract

him/her from the analysis task and that has the additional disadvantage that (s)he might miss events while moving the camera.

Automatic camera control methods could provide solutions to such problems, thus relieving the user from the burden of manual camera placement, selection, and control. However, most methods that have been proposed in the literature are designed for sequential dialogue-like events involving at most two or three characters and are thus not suited to situations that include several parallel events involving many characters. Indeed, none of these solutions have been adopted for emergency simulations.

In this paper, we present a novel and fully automated, real-time camera control system for emergency simulations that is able to monitor interesting events and present them to a viewer. We propose to organize such a system into two conceptual modules: a *Camera Operator* and a *Director*. The *Camera Operator* is based on extending a recent virtual camera computation approach [4] to calculate, whenever needed, a virtual camera that aims at visualizing the maximum number of currently occurring events of interest. The *Director* then analyzes the virtual cameras that are computed by the *Camera Operator* and chooses which camera to use and when to use it to visualize simulation events to the viewer. To illustrate how the system operates in practice, this paper applies it to a complex case in the domain of aviation safety. However, the system is not limited to aviation and could be utilized in other emergency domains.

The paper is organized as follows. In Section 2, we briefly review past work on computer-based emergency simulations and automatic camera control and motivate the need for the proposed approach. In Section 3, we describe the proposed camera control system, and in Section 4, we apply the system to a full

62 aircraft evacuation scenario that reproduces the main aspects of
63 a well-known recent accident. Finally, in Section 5, we con-
64 clude the paper and outline future research directions.

65 2. Related Work & Motivations

66 2.1. Computer-based Emergency Simulations

67 Computer-based emergency simulations are increasingly used
68 for a variety of purposes, including planning, prediction of out-
69 comes, accident investigations, and training. In particular, emer-
70 gency evacuations have received considerable attention in the
71 literature. Gwynne et al [5] reviewed 22 evacuation models and
72 classified them into three main categories: optimization, simu-
73 lation and risk assessment. EXODUS is a well-known evacua-
74 tion model that was successfully applied to analyze both build-
75 ing [6] and mass-transport [7] evacuations. The system includes
76 specialized modules to model very specific aspects such as (i)
77 the characteristics of occupants (e.g., age, gender, and physi-
78 cal disabilities), (ii) their movements and behaviors, and (iii)
79 the physiological impact of toxicity due to smoke. A variant
80 of EXODUS, called airEXODUS [8], is specifically tailored to
81 aircraft evacuation.

82 In general, EXODUS and the other evacuation models at-
83 tempt to precisely compute the values of several variables to
84 predict an evacuation outcome or analyze a real case, but they
85 do not focus on real-time interaction (e.g., interactions that dy-
86 namically affect an evolving simulation) and have limited vi-
87 sualization features (e.g., 2D maps or simplified 3D models).
88 In particular, the EXODUS system can be used with vrEXO-
89 DUS, which is a 3D visualizer of the simulations that operates
90 as a graphics post-processor of previously generated simula-
91 tions. The Glasgow Evacuation Simulator [9] introduced the
92 possibility of opening or closing routes in real time to test dif-
93 ferent evacuation paths. Moreover, the simulator supports the
94 visualization of an evacuation using CAD-CAM 3D models of
95 buildings, but occupants are represented only by colored cylin-
96 ders.

97 In addition to advancing the simulation domain, improv-
98 ing the realism of graphics and real-time interaction with the
99 simulator would extend the application of evacuation models to
100 training, thus allowing trainees to learn by directly interacting
101 with virtual objects and characters. Moreover, with the help of
102 3D animations, trainees could virtually experience emergency
103 scenarios that are difficult, expensive and dangerous to repro-
104 duce in the real world, thereby getting a better understanding of
105 complex scenarios and cause-effect relationships [10, 11]. Sys-
106 tems that employ realistic 3D graphics consider various emer-
107 gency scenarios such as car accidents with fire and toxic gas
108 propagation in road tunnels [1], smoke hazards in subway sta-
109 tions and schools [12], fire drills in buildings [13], and evacua-
110 tions of airports [3] and nuclear facilities [2].

111 2.2. Automatic Camera Control

112 Current approaches to the visualization of the 3D simula-
113 tions discussed in the previous section are based on either plac-
114 ing virtual cameras in the scene at modeling time, and switching

115 between them at run time, or manually controlling a moving
116 camera at run time. However, depending on the spatial com-
117 plexity of the scenario, even a very large number of cameras or
118 a very skilled manual control will fail to correctly frame certain
119 events, e.g., because of dynamic occluders. Moreover, manu-
120 ally placing the cameras can involve a considerable modeling
121 effort, which in general has to be repeated for each simulation,
122 and manual camera control in real time imposes a cognitive load
123 on the viewer that can distract him/her from the analysis task.
124 Many emergency simulations involve hundreds (or even thou-
125 sands, as in simulations of the 9/11 attack [14]) of independent
126 characters and many different types of events that are occurring
127 in parallel over an area that could be very large. As a result, it
128 is very hard to select and visualize all of the relevant details of
129 such emergencies with the camera control approaches of cur-
130 rent simulators.

131 Automatic camera control methods could offer a method of
132 addressing these issues. In the following, we analyze the main
133 aspects that an automatic camera control system must consider
134 to present a simulation. For each aspect, we briefly discuss the
135 state of the art and illustrate why current approaches are not
136 adequate for emergency simulations.

137 The first fundamental aspect is how to find virtual cameras
138 that ensure the visibility of events of interest. Current auto-
139 matic control approaches can be organized into two main cate-
140 gories: approaches that search for virtual cameras anywhere in
141 the scene and that can consider an arbitrary number of targets
142 [4, 15, 16], hereinafter called *global solvers*, and approaches
143 that focus on ensuring the continuous visibility of one [17, 18]
144 or a few [19, 20] dynamic targets and that search only in a re-
145 gion around a current camera. In both approaches, visibility
146 is typically defined in terms of a combination of various vi-
147 sual properties such as target screen size, occlusion, and angle
148 from which the target is observed. In emergency simulations,
149 events might occur anywhere in the scene; therefore, the abil-
150 ity to find virtual cameras anywhere in the scene is substan-
151 tially more important than ensuring continuity in visualizing
152 the simulation. Unfortunately, most global solvers typically
153 suffer from performance issues because they rely on stochas-
154 tic optimization strategies (e.g., population-based algorithms)
155 to sample the search space. An exception is a recent proposal
156 by Ranon and Urli [4], who introduced more effective candi-
157 date camera initialization and evaluation strategies whereby a
158 single virtual camera can be computed in tenths of milliseconds
159 (instead of hundreds) in quite complex scenes.

160 The ability to find cameras that can frame various current
161 events of interest is only the first step toward the broader goal of
162 conveying meaning (or at least making it inferable) to a viewer.
163 This topic has been the subject of several research papers, e.g.,
164 [21, 22, 23, 24], that focus on narrative events and mimic the
165 language of films by encoding cinematographic rules such as
166 typical shots and continuity editing. However, such approaches
167 are limited to film dialogue-based interactions among two or
168 three characters and to consider one event at each time. For ex-
169 ample, the *Virtual Cinematographer* [23] and the FILM system
170 [24] are able to film events in real time by selecting among a
171 set of *idioms*. An idiom contains information about the number

172 of targets, shot types and, in the *Virtual Cinematographer*, the
 173 timing of transitions between shots to best communicate events,
 174 such as three virtual actors conversing, as they unfold. Camera
 175 placement is selected among a few pre-encoded, idiom-specific
 176 alternatives, e.g., depending on the targets' visibility. However,
 177 there is no guarantee that, in a spatially complex environment,
 178 any of the alternatives will provide a suitable framing of tar-
 179 gets. Lino et al [21] improved upon the two above-mentioned
 180 systems by considering a narrative event and computing a set
 181 of *director volumes*, i.e., volumes in the scene that encode both
 182 shot type and visibility information for the considered event.
 183 Then, they searched the director volumes for optimal virtual
 184 cameras that guarantee continuity when cutting between cam-
 185 eras and selected the best virtual camera based on style ele-
 186 ments. In this approach, the visibility computations are per-
 187 formed in 2D (therefore, they are not applicable to multi-level
 188 scenes or small objects) and do not consider dynamic occlud-
 189 ers such as other characters in the scene. In summary, current
 190 systems based on cinematographic rules work well in situations
 191 where spatial complexity is limited, the scene is mostly static,
 192 and camera control targets consists of mainly two or three char-
 193 acters that are engaged in dialogue-like events. Moreover, such
 194 systems can typically consider only one event at a time. Due to
 195 these limitations, they are poorly suited to emergency simula-
 196 tions.

197 A system that better addresses the needs of emergency sim-
 198 ulations was proposed by Galvane et al [25], who focused on
 199 presenting events that occur in crowd simulations. Their system
 200 relies on Reynolds' model of steering behaviors to control and
 201 locally coordinate a collection of camera agents in real time in
 202 a manner similar to a group of reporters. Camera agents can be
 203 either in a scouting mode, thereby searching for relevant events
 204 to present, or in a tracking mode, thereby following one or more
 205 unfolding events. The system was tested using a crowd simu-
 206 lation with 100 virtual characters in an exterior environment,
 207 where it provided a good coverage of events (mainly measured
 208 as the ratio between observed versus missed events). Com-
 209 pared to our method, their camera control approach has vari-
 210 ous advantages and disadvantages. The main advantage of their
 211 method is that it directly provides smooth camera motions in
 212 contrast to the static cameras that we use, which is a feature that
 213 might be preferable when the result should exhibit cinematic-
 214 like qualities. Moreover, the approach of using a population of
 215 cameras naturally enables the simultaneous coverage of events
 216 that occur at different locations in the simulation or the cover-
 217 age of the same event from different perspectives. The disad-
 218 vantage of moving cameras through smooth motion (no "tele-
 219 port") is that it might take some time before a camera is able to
 220 reach the position where an event occurs. Because this duration
 221 depends on where the cameras are at the moment of event oc-
 222 currence, camera movement time is not predictable. Increasing
 223 the number of cameras might help, but this would also increase
 224 the computational complexity (their paper reports 15 fps for
 225 30 cameras). Moreover, their approach provides a better per-
 226 formance and was demonstrated using exterior scenes, where
 227 cameras can observe and easily detect events that occur without
 228 occlusions. For interior or mixed interior/exterior scenes that

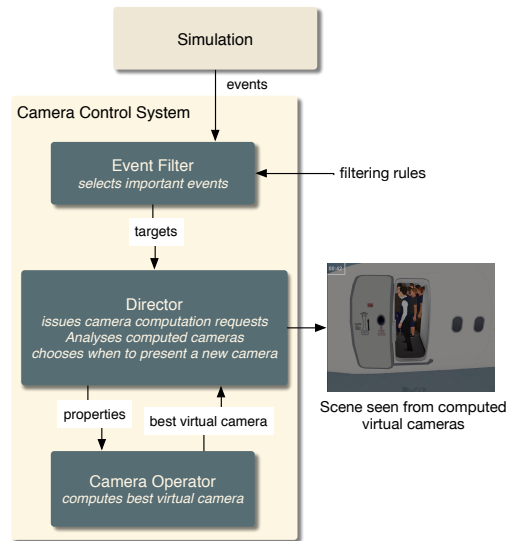


Figure 1: Overview of our system.

229 are typical of emergency evacuations, their camera computa-
 230 tion methods would likely take more time than in purely exter-
 231 ior scenes to discover events because they would not be able to
 232 detect an event unless it is visible. Finally, their approach does
 233 not consider how to select the camera to present events to the
 234 viewer.

235 3. The proposed system

236 In this section, we present a system that can monitor events
 237 from a simulation, select interesting ones based on user prefer-
 238 ences, and present the events to a viewer. Events refer to objects
 239 in the 3D scene, which are considered as targets that should be
 240 visualized. Our system can operate in real time (i.e., while the
 241 simulation updates the 3D scene) without any assumption or
 242 pre-processing of the spatial environment or the behavior and
 243 shape of 3D objects. The system does not attempt to present
 244 events using a cinematic language (e.g., preserving continuity
 245 between cuts); the system uses only static virtual cameras be-
 246 cause they are sufficient for monitoring a simulation.

247 To illustrate how the system works in practice, we will ap-
 248 ply it to the domain of aviation safety. However, the system is
 249 not limited to aviation and can be reused in other emergency
 250 domains.

251 An overview of our system is provided in Figure 1. The
 252 simulation sends a stream of all events that occur to the camera
 253 control system. The events are then filtered in real time (on the
 254 basis of user-provided *filtering rules*) to select the ones that are
 255 relevant to the current viewer. For each event that passes the
 256 filtering phase, the system extracts *targets*, i.e., objects in the
 257 simulation that are involved in the event. Every few seconds,
 258 the *Director* module takes the list of *targets* that have been ex-
 259 tracted so far and asks the *Camera Operator* module to com-

260 pute a virtual camera that visualizes the targets by creating a
261 list of properties that the desired virtual camera should satisfy.
262 The *Camera Operator* then tries to determine the virtual camera
263 that best satisfies the request and returns the result to the
264 *Director*, which evaluates the result and decides if and when to
265 present it to the viewer by activating a transition from the current
266 camera to the new camera. In the following, we describe in detail
267 the major activities performed by our system.

268 3.1. The Event Filter Module

269 For each event that occurs, the running simulation sends
270 an event description to the camera control system. An event
271 description is a (*subject*, *action*, *object*) triplet where

- 272 • *subject* is the acting object in the simulation event (e.g.,
273 "*Flight Assistant 1*");
- 274 • *action* is a textual description of the action performed by
275 the subject (e.g., "*starts opening door*"); and
- 276 • *object* is the object in the simulation that is affected by
277 the action (e.g., "*door 1L*");

278 As explained in Section 1, viewers are typically interested
279 in a subset of events, and the subset varies according to the pur-
280 poses for which a simulation is run. To select the subset of
281 events, the viewer provides a list of strings, each of which can
282 be the name of an object in the simulation or (part of) an ac-
283 tion. *Filtering rules* then perform substring matching between
284 the list of strings and the stream of events output from the sim-
285 ulation. For example, the list of strings ("*Flight Assistant 1*",
286 "*door*") will match all events where *Flight Assistant 1* or any
287 *door* are involved. Matching events are then parsed, and simu-
288 lation objects contained in them are inserted in the *targets* list,
289 which is accessed by the *Director* module.

290 Because a simulation might, at times, not generate events
291 that match, the camera control system allows the viewer to spec-
292 ify a set of targets that should be framed in the absence of inter-
293 esting events. For example, one could specify the entire scene
294 if (s)he wants the camera control system to search for global
295 overview shots when no events match his/her interests.

296 3.2. The Camera Operator Module

297 The *Camera Operator* module is based on a recent open-
298 source declarative virtual camera computation library devel-
299 oped by Ranon and Urli [4], which is able to compute in a
300 given amount of time the virtual camera that best satisfies a list
301 of visual properties. The visual properties can express desired
302 values of the size (area, width or height), visibility, camera an-
303 gle and on-screen position for any choice of objects in the 3D
304 scene. From an input list of visual properties, the library first
305 builds a function that returns a numeric value indicating to what
306 extent a given virtual camera satisfies the properties. Then, a
307 solver based on Particle Swarm Optimization [26] iteratively
308 searches the 3D scene for the virtual camera that maximizes the
309 satisfaction function. The library works with any type of scene
310 or object and does not require any preprocessing of the scene;

311 the library relies on the 3D rendering engine to obtain infor-
312 mation about the bounding volumes of objects and to perform
313 ray casting queries to measure visibility. A solution can be re-
314 turned in any amount of time, although in complex scenes, ad-
315 ditional computation time will generally translate into a better
316 solution (i.e., greater satisfaction of visual properties). We refer
317 the reader to [4] for a detailed explanation of the optimization
318 approach¹.

319 In this paper, we need to define a set of properties that char-
320 acterize any virtual camera that can frame a specific list of tar-
321 gets, thereby making them prominent in the images rendered
322 from the camera and ultimately allowing a viewer to understand
323 the events that involve them. Our proposal is to use the follow-
324 ing properties for each target:

- 325 • the screen area of the *target* should preferably be at least
326 10% of the screen size when we only have one target; in
327 this way, it will be the main or one of the main subjects
328 in the displayed image so that the viewer can easily recog-
329 nize it, and the viewer will also see objects around it,
330 to understand its position in the scene. When there are
331 more targets, the value is divided by their number;
- 332 • the *target* should be fully visible (no other objects oc-
333 cluding it);
- 334 • the *target* should be framed as close as possible to the
335 center of the screen (so that the viewer's attention is drawn
336 to it);
- 337 • the *target* should be viewed from the front. The front is
338 defined based on the category of the object. For example,
339 in the case of a character, this means being able to see the
340 character's face.
- 341 • the *target* should be viewed from a medium to high an-
342 gle (we want to avoid viewing the target from too high
343 or too low because in such cases, it could be difficult to
344 understand what the target is doing).

345 Note that certain properties are more important than others.
346 For example, the visibility of a target is clearly more important
347 than framing it in the center of the screen because it is certainly
348 preferable to be able to see a target, even close to the screen
349 edge, than to not be able to see it because of occlusions caused
350 by other objects.

351 In the adopted virtual camera computation library, the above
352 requests are implemented by a `Size`, `Occlusion`, `Framing`,
353 and two `Angle` properties with the target as first argument. The
354 second argument of each property is a satisfaction function that
355 returns values in [0,1] (where 0 means no satisfaction and 1
356 means full satisfaction) expressed as a linear spline. Table 1
357 presents the visual properties that we have associated to each
358 target and their corresponding satisfaction functions. For exam-
359 ple, for the `Size` property, we have defined the spline in terms
360 of the points (0,0), (0.05, 0.01), (0.08, 0.8), (0.1,1) and (1,1),

¹Source code of the library is available at <http://bit.ly/lwdBOqq>

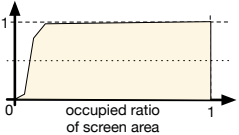
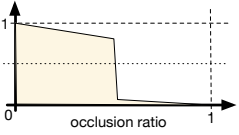
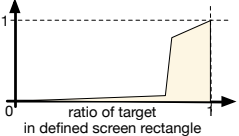
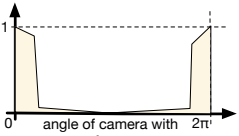
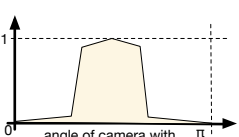
Property Type	Semantics	Weight	Satisfaction Function
Size	the target object should cover at least 10% of the screen area, or less if there are multiple target objects	2.5	
Occlusion	the target should not be occluded by other objects	4.0	
Framing	the target object should be framed inside a screen rectangle with minimum and maximum corners equal to (0.2,0.2) and (0.8, 0.8), respectively, in viewport coordinates	1.0	
Angle	camera in front of the object	1.0	
Angle	camera parallel or slightly above object	1.5	

Table 1: Properties defined for each target object to compute a virtual camera. Weights reflect relative importance of properties and have been determined empirically. Slight variations of weights would not alter substantially the result.

361 where the x value in the function is the ratio of the screen area
362 that is occupied by the rendered target and the y value is the
363 corresponding satisfaction value. The weight of each property
364 (shown in the table) is a number that reflects the relative im-
365 portance of the property compared to the other properties. For
366 example, the weight of the `Occlusion` property is four-times
367 larger than the weight of the `Framing` property. Because the
368 satisfaction of a virtual camera is defined as a weighted sum of
369 all the property satisfaction functions, this means that in cases
370 when both framing and visibility cannot be guaranteed, the op-
371 timization process will prefer visibility. Weights have been em-
372 pirically determined by running a few simulations, and slight
373 variations do not alter substantially the result.

374 In certain situations, it might be preferable to use other sets
375 of properties. For example, the viewer might be more interested
376 in visualizing an overview of passengers exiting from the doors
377 on a plane. In this case, the goal is not to obtain virtual cameras
378 that are sufficiently close to recognize a character but to derive
379 virtual cameras that visualize groups of characters in different
380 positions in the scene. This can be expressed by requiring, for

381 each target, a minimum screen area that is much lower than 10%
382 and to soften requirements concerning angle (so that, for exam-
383 ple, top-down virtual cameras also satisfy the requirements). In
384 our system, viewers can choose (and even modify their choice
385 while the simulation is running) whether to promote *target rec-*
386 *ognizability* (i.e., the *Camera Operator* will try to frame targets
387 at a close distance using all of the above properties) or *event*
388 *coverage* (i.e., the *Camera Operator* will try to frame targets
389 at a greater distance if this is necessary to capture more targets
390 using the more relaxed properties described above). In the last
391 case, the properties are modified as follows: for each target, the
392 minimum screen area becomes 0.5%, the `Angle` property that
393 considers the target front vector is removed (so that it is equally
394 satisfying to frame the target from behind) and the `Angle` prop-
395 erty that considers the target up vector considers as satisfying
396 any angle from 0 to 45 degrees.

397 As with all virtual camera computation approaches based
398 on optimization, there is no guarantee that the best returned vir-
399 tual camera will satisfy all of the given properties. First, such
400 a camera might not exist; e.g., consider a case in which we are

401 simultaneously interested in two targets that are located in two
402 opposite zones of an aircraft: the pilot's cockpit and the rear
403 galley. These situations are not unlikely because, in principle,
404 the simulation might compute events that occur simultaneously
405 in completely different locations. In general, our system will try
406 to find the virtual camera that frames more targets because this
407 corresponds to greater satisfaction. In the example of the two
408 targets in opposite zones of an aircraft, the system will have
409 to choose one of the targets and miss the other (which could
410 be framed later). A more subtle issue is the case where only
411 certain properties of a target can be satisfied. For example, it
412 might be impossible to find a camera that guarantees both the
413 required size and (at least partial) visibility. In this case, while
414 the returned camera satisfies certain target properties, it may
415 not allow the viewer to understand the events that involve the
416 target. More generally, due to the limited time available for
417 virtual camera computation and the stochastic nature of search,
418 the *Camera Operator* might at times be unable to find a virtual
419 camera that satisfies all properties or even all of the properties
420 for some targets even if such a virtual camera exists. In general,
421 as the geometrical complexity of the scene and/or the number
422 of targets increase, this type of issues are more likely to occur.
423 An increase in geometrical complexity typically translates into
424 more time required to explore the scene in search of a camera
425 that satisfies the visibility properties. A larger number of prop-
426 erties increases the time required to evaluate the satisfaction
427 of virtual cameras during the search process. We address all
428 these issues by evaluating the virtual camera that is computed
429 by the *Camera Operator* before using it to visualize events to
430 the viewer.

431 3.3. The Director Module

432 The *Director* module manages the entire camera control
433 process. This module decides which camera is shown to the
434 viewer, how and when to transition to a new camera, issues vir-
435 tual camera computation requests to the *Camera Operator* and
436 evaluates the returned virtual camera. The *Director* module is
437 executed at regular time intervals (0.2 seconds), and its opera-
438 tion is schematized in Figure 2.

439 When it is time to change the camera to be shown to the
440 viewer, the *Director* takes the current *targets* list from the *Event*
441 *Filter* module, computes the list of properties, issues a virtual
442 camera computation request to the *Camera Operator*, and stops
443 its current execution. If instead this operation was performed
444 during the previous execution, the *Director* would take the vir-
445 tual camera that meanwhile has been computed by the *Camera*
446 *Operator*, evaluates it and decides if the camera should be used.
447 In such a scenario, the *targets* list is emptied.

448 The evaluation of the virtual camera returned by the *Camera*
449 *Operator* considers which of the targets are effectively framed,
450 i.e., the involved events are recognizable. We define a tar-
451 get as effectively framed by a virtual camera if its *Size* and
452 *Occlusion* properties have a minimum satisfaction value of
453 0.5 and 0.3, respectively, out of 1. This corresponds to the tar-
454 get being half of the preferred minimum screen area and half
455 visible. For the other properties, we rely on the virtual cam-
456 era computation process to maximize the satisfaction, but we

457 also accept virtual cameras that do not frame certain targets
458 in the screen center or with the required angle because these
459 requirements could likely be difficult or impossible to satisfy
460 for multiple targets simultaneously. Therefore, for each virtual
461 camera, we compute two lists of targets: targets that are effec-
462 tively framed (*framed targets*) and frames that are not effec-
463 tively framed (*missed targets*). The evaluation has a negligible
464 computational cost because it was previously performed dur-
465 ing the search process. A virtual camera is deemed to be good
466 for the viewer when its *framed target* list contains at least one
467 target. If a virtual camera is not good, it is discarded. When
468 the *Camera Operator* fails to find a good camera in the allowed
469 time frame, the *Director* immediately issues a new virtual cam-
470 era computation request with one target removed from the *tar-*
471 *get* list (preferably one of the targets that are already framed by
472 the current camera or a random target). The result of the virtual
473 camera computation request will be available for evaluation in
474 the next execution of the *Director* module. Targets in the vir-
475 tual camera *missed target* list, if present, will be considered as
476 targets for the next virtual camera computation if no interesting
477 events are detected in the next camera update cycle.

478 A good virtual camera, before being used, is compared to
479 the current camera. It is not always necessary to transition to a
480 new camera; there are times in which new interesting events in-
481 volve targets that the current virtual camera is already framing,
482 or a newly computed virtual camera does not provide signifi-
483 cantly more information than does the current one. For this rea-
484 son, before changing the current virtual camera, we compare
485 it with the new candidate camera. If the new virtual camera
486 frames more or different targets compared to the current cam-
487 era, we transition to it. If they frame the same targets, then
488 we transition to the new camera only if its satisfaction value is
489 greater than the current value by at least 5%. If the newly found
490 camera frames only a subset of the targets that the current cam-
491 era is framing, we maintain the current camera.

492 The frequency of transitions to a virtual camera that visual-
493 izes new events is a critical choice. To maximize the coverage
494 of events, we should compute a new virtual camera and possi-
495 bly transition to it as soon as the event passes the filtering stage.
496 However, in simulations of emergencies wherein many events
497 can occur in a short amount of time, this could result in mul-
498 tiple virtual camera changes per second, which would make it
499 impossible for the viewer to understand what is occurring.

500 General rules of cinematography dictate that static shots
501 should last between 2 and 10 seconds. In our system, the viewer
502 can set the minimum time between virtual camera transitions.
503 In testing our system on aircraft accident scenarios, we have
504 empirically noted that a time of 3-4 seconds is a good compro-
505 mise between event coverage and comprehension.

506 Transitions are currently implemented as straight cuts. While
507 this solution has the disadvantage that it takes a bit of time for
508 viewers to understand the camera changes, allocating time for
509 the camera to transition from the old location to the new loca-
510 tion could make viewers miss events. However, when the new
511 camera is close to the current camera in terms of position and
512 orientation, a smooth transition may be better for the viewer.

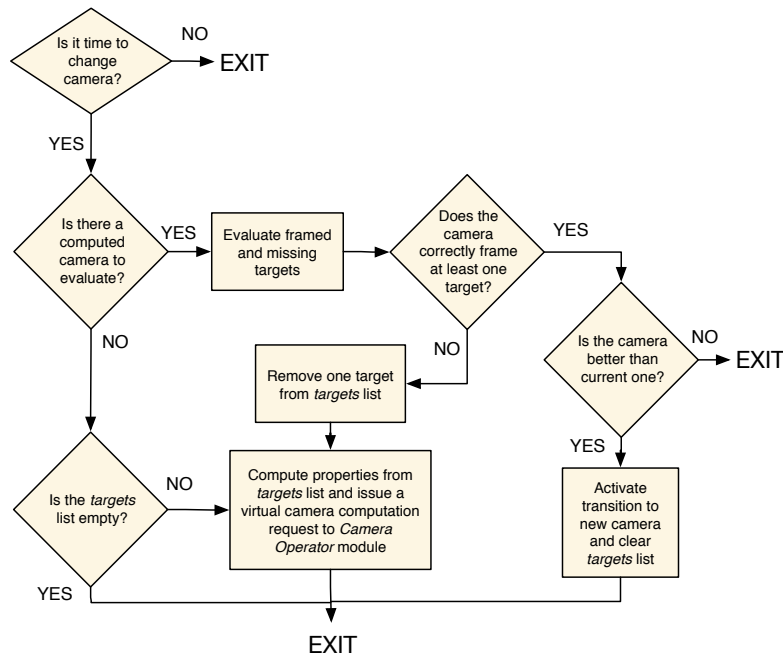


Figure 2: Functioning of the *Director* module.

513 4. Results

514 We have extensively tested our system using simulations of
 515 different types of aircraft emergencies. In particular, the ex-
 516 amples presented in this section concern a full aircraft water
 517 landing (ditching, in aviation terminology) and evacuation sce-
 518 nario. More precisely, we employ an accurate 3D reconstruction
 519 of an Airbus 320 [27], one of the most common aircraft
 520 types in service. The reproduced accident is very similar to the
 521 accident involving US Airways flight 1549 [28]: a few minutes
 522 after take-off, the aircraft suddenly loses thrust in both engines
 523 due to a severe strike with a flock of large birds and is forced
 524 to ditch because the lack of thrust makes it impossible to reach
 525 nearby airports.

526 The 146 virtual passengers in the simulation can perform
 527 several autonomous tasks, which include the following: (i) fas-
 528 tening seat belts as soon as the airborne plane shows signs of
 529 instability, (ii) maintaining the brace position during ditching
 530 until the plane comes to a stop, (iii) reaching for the nearest
 531 exit, (iv) locating an alternative exit in the presence of exits that
 532 cannot be used (e.g., in the following examples, the rear exits
 533 are not usable because they are below the water level), (v) open-
 534 ing overwing doors, (vi) exiting the aircraft using a level exit or
 535 an overwing exit, and (vii) going toward the bottom of a slide
 536 raft and sitting on it. Moreover, the simulation includes three
 537 virtual flight assistants that can perform three additional tasks:
 538 (i) open floor level doors, (ii) order passengers to stand back
 539 until a raft is fully inflated, and (iii) block unusable exits and
 540 redirect passengers. In the examples presented in this section,

541 two flight assistants help passengers at the front exits, while the
 542 other attendant blocks the two unusable rear doors and redirects
 543 passengers to the front and overwing exits until all passengers
 544 are away from the flooded rear galley.

545 Each passenger and flight assistant has a unique name (e.g.,
 546 "Passenger 113" or "Flight Assistant 1") in the simulation, and
 547 all aircraft-relevant parts are labeled (e.g., "door 1L" or "door
 548 2L"). These names and labels are used as subjects and/or ob-
 549 jects in the event triplets. The event actions concern all of the
 550 tasks described above as well as changes in states of the aircraft
 551 doors (e.g., closing and opening) and slides (e.g., inflating).

552 The total number of events in each of the following exam-
 553 ples is 1829. The first 590 events (e.g., fastening seat belts
 554 and assuming a brace position) occur in the 4 minutes and 20
 555 seconds during which the aircraft is airborne. The other 1239
 556 events occur during the evacuation, which lasts approximately 2
 557 minutes and 30 seconds. In the following examples, we will fo-
 558 cus on the evacuation because this phase contains a large num-
 559 ber of events in a limited amount of time; thus, it presents a
 560 greater challenge to the camera control system. Note that, due
 561 to the stochastic nature of particle swarm search, the system can
 562 generate different cameras in different runs even if the simula-
 563 tion and its events are identical.

564 We describe three examples of system use. For each exam-
 565 ple, we describe the scenario and provide sample screenshots.
 566 In addition, to enable the reader to see first-hand the actual out-
 567 put of the system, we have included a video as additional paper

568 materials². Both the examples presented in this Section and the
569 video use a frequency of camera transition of 4 seconds. In
570 Section 4.1, we analyse the performance of the system in these
571 scenarios.

572 The first example (at minutes from 00:18 to 02:03 in the
573 accompanying video) considers the perspective of flight assis-
574 tant training, in which trainers and trainees are highly interested
575 in observing the behavior of the crew. Therefore, we spec-
576 ify “flight assistant” as a matching string in the filtering rules.
577 The *Event Filter* module will then discard all events that do
578 not match this string while selecting all evacuation events con-
579 cerning flight assistants (30 in our example). As a result, the
580 *Camera Operator* module will receive requests to frame one,
581 two, or three flight assistants simultaneously (depending on the
582 timing of events) as well as the object that they could interact-
583 ing with (e.g., doors). Because it is important to frame the flight
584 assistants at close distances in this example, we set the system
585 to prefer target recognizability over event coverage. Figure 3
586 shows six of the 30 cameras that were computed and used by
587 our system using these settings during the entire simulation.

588 More precisely, immediately after the impact, all three flight
589 attendants simultaneously stand up. Because two of them are
590 at the front exits and one is at the rear exits, it is impossible to
591 simultaneously frame all of them, and the system finds a camera
592 showing the two at the front, as shown in Figure 3a. One of
593 the flight assistants at the front exits is the first to reach and
594 open a door, as shown by the camera in Figure 3b. When the
595 second flight assistant at the front exits reaches and opens the
596 other front door, both flight assistants order passengers to stand
597 back until the slides are fully inflated. In this case, our system
598 finds a camera that frames both subjects (Figure 3c). When a
599 front slide is fully inflated, the system shows the nearby flight
600 attendant stepping aside and indicating the way to passengers,
601 as shown in Figure 3d. In contrast, the flight assistant at the rear
602 exits is sending passengers away because water is entering the
603 rear galley (Figure 3e). Only when all passengers have left the
604 rear galley can the flight attendant move forward (Figure 3f) and
605 exit the aircraft (Figure 3g). When all passengers have exited
606 the aircraft, the other flight assistants can exit (Figure 3h).

607 The second and third examples consider the perspective of
608 an aircraft designer or an accident investigator, who are inter-
609 ested in observing how, where and when passengers and crew
610 exit an aircraft in an accident. In this case, we specify “exit” as
611 a matching string in the filtering rules. The *Event Filter* mod-
612 ule then selects all exit events (149 in our case, one for each
613 of the 146 passengers plus three for the flight assistants). Exit
614 events begin at the exact instant a passenger exits the plane and
615 last about two seconds. As a result, the *Camera Operator* mod-
616 ule will often receive requests to simultaneously frame many
617 passengers (depending on the timing of exit events) and mainly
618 focus on doors. In this case, different viewers can be interested
619 in observing the exit behaviour with different priorities: some
620 viewers may be interested in watching passengers and exits at
621 a close distance, while other viewers may be more interested in

622 understanding the egress as a whole. Therefore, in our second
623 example (at minutes from 02:05 to 03:29 in the accompanying
624 video), we set the system to prefer target recognizability, while
625 in the third example (at minutes from 03:30 to the end in the ac-
626 companying video), we set the system to prefer event coverage.

627 If the system is set to prefer target recognizability, when the
628 first passengers begin to exit on the left wing, the camera cor-
629 rectly focuses on them (Figure 4a). A few seconds later, passen-
630 gers start to use the right overwing exits. In this case, there is
631 no camera that can simultaneously frame all of the overwing ex-
632 its and the exiting passengers while preserving recognizability;
633 therefore, the Director module can choose to continue showing
634 passengers exiting on the left wing or switch to the right wing.
635 Figure 4b shows the second choice. When the front right raft
636 is fully inflated and passengers start using the front right exit,
637 our system can find cameras that frame both front and overwing
638 right exits (Figure 4c). When all front and overwing exits are
639 available, at each camera update, the system computes the po-
640 sition and angle that maximize the properties shown in Table 1
641 for the highest number of targets (Figure 4d and 4e). In partic-
642 ular, when only one exit is used, the system can compute a new
643 camera to frame only it (Figure 4f).

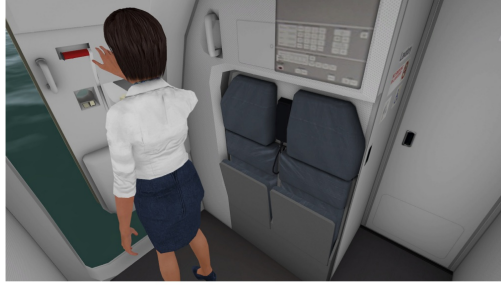
644 If event coverage is preferred instead, the system will find
645 more distant cameras that can frame more targets. More pre-
646 cisely, at the beginning of the simulation, when there are pas-
647 sengers only exiting from the left overwing exits, the camera
648 will specifically focus on them, as in Figure 5a, but when pas-
649 sengers begin to use the right overwing exits, the system will
650 try to find a camera that can simultaneously frame passengers
651 at all of the overwing exits, as in Figure 5b. When the right raft
652 is inflated and passengers start to use it, the system computes
653 a camera that focuses on exits at the right side but continues to
654 frame passengers exiting from the left overwing exits (Figure
655 5c). Finally, when the left raft is also available, the cameras will
656 try to simultaneously frame all used exits, as in Figure 5d. As
657 in the previous example, when only a subset of exits is used at a
658 particular moment in the simulation, the system will compute a
659 more focused camera, as in Figure 5e, but the different settings
660 will also include cameras that simultaneously cover events that
661 are occurring at the two opposite sides of the aircraft (Figure
662 5f), despite partially reducing recognizability.

663 While an extensive and formal evaluation with domain ex-
664 perts has not yet been performed, we have informally tested the
665 system by using the videos it creates as a means of communica-
666 tion with aviation professionals (researchers and pilots) as well
667 as individuals who are unfamiliar with aviation (students and
668 researchers in other domains). In each case, we first informed
669 the viewer about the general goals of the videos (i.e., the kind of
670 events that the camera control system was instructed to frame).
671 Then, we showed the video without any comment or verbal
672 explanation. Finally, we discussed about the aircraft accident
673 and evacuation depicted by the video to check if there were
674 any comprehension issues or doubts concerning the important
675 events. From this purely informal experience, the output pro-
676 duced by the system appears to be effective: the videos were
677 clearly understood without ambiguities and the events are ef-
678 fectively presented. A possible issue that emerged is that some-

²The video is also available at <http://youtu.be/DJq87oasil8>



a)



b)



c)



d)



e)



f)



g)



h)

Figure 3: Various cameras from the ditching simulation when the filtering rules specify to match "flight attendant" events and the preferences are set to target recognizability.



a)



b)



c)



d)



e)



f)

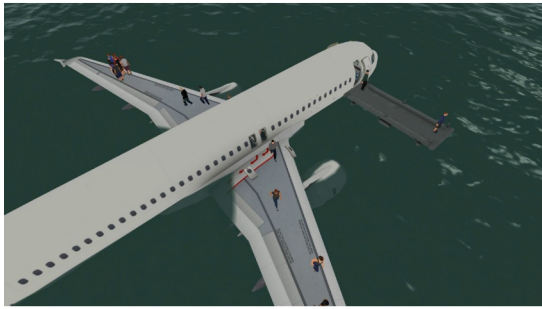
Figure 4: Various camera shots from the ditching simulation when the filtering rules specify to match "exit" events and the settings prefer target recognizability over event coverage.



a)



b)



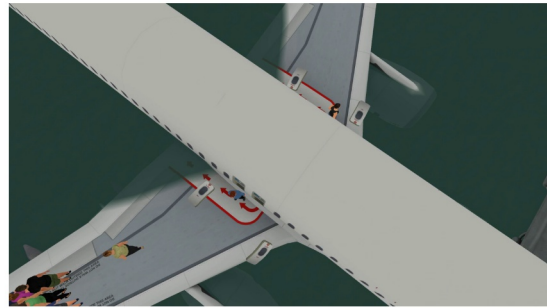
c)



d)



e)



f)

Figure 5: Various camera shots from the ditching simulation when the filtering rules specify to match "exit" events and the settings prefer event coverage over target recognizability.

679 times the change in camera position and orientation required a
680 few instants for the user to reorient herself, although this issue
681 concerns mainly viewers who are not familiar with the detailed
682 internal and external structure of an aircraft. The system proved
683 also very useful while working on the simulation visualization,
684 because it provided a way for developers to focus on specific
685 parts of the simulation and check for graphical glitches.

686 4.1. Implementation and Performance

687 The scenarios presented above are implemented in C# using
688 the Unity 4 game engine [29]. The camera control system, as
689 well as the simulation, are implemented in C# as a Unity scripts,
690 and due to Unity limitations, run on the same thread on the CPU
691 (i.e., they cannot run in parallel). Since computing and render-
692 ing the simulation is already demanding on the CPU, the com-
693 putational cost of the camera control system should be as low
694 as possible. By far, its most expensive activity is the computa-
695 tion of cameras in the *Camera Operator* module, whose allotted
696 time, as explained in Section 3.2, is a parameter that can be set.
697 However, in deciding its value, one faces two contradicting ob-
698 jectives. On the one hand, by allowing more time to the *Camera*
699 *Operator*, we increase the probability of finding more satisfying
700 solutions (i.e., cameras that better frame events or frame more
701 of them); on the other hand, more time means, in CPU-bound
702 applications like our simulation, decreasing the frame rate. For
703 example, in our simulation, if we set the time available to the
704 *Camera Operator* to 30 milliseconds, this means that each time
705 a camera needs to be computed, the frame rate will drop drasti-
706 cally as the CPU cost for each simulation frame (without the
707 camera control) is already, on average, around 25 milliseconds.

708 To help with this issue, we can split the computation of a
709 camera among a few consecutive frames to limit its impact on
710 frame rate, with the only drawback of delaying the presentation
711 of the result by a few frames. Figure 6 illustrates the perfor-
712 mance, in the scenarios described in the previous Section, of
713 three different choices about the time for computing a new cam-
714 era: 7 milliseconds in one frame, 14 milliseconds equally split
715 among two frames, and 21 milliseconds equally split among
716 three frames. To measure the performance, we use the num-
717 ber of framed events, as the average frame rate is similar in all
718 cases. By looking at the box plots, it is clear that there is a gen-
719 eral increase in the number of framed events, for all quartiles,
720 both by going from 7 milliseconds to 14, and from 14 to 21.
721 The increase is more notable in the exit scenarios, which are
722 more complex in terms of average number of targets to frame.
723 Note also that, in the exit scenario, by preferring event cover-
724 age over target recognizability, we greatly increase the median
725 number of framed events (by around 65% in the condition with
726 21 milliseconds).

727 We tried also a fourth condition with 28 milliseconds split
728 in four frames, but it did not result in any significant increase in
729 performance. As explained in the previous Section, the reason
730 is that in all scenarios events happen very often in parallel and
731 in different parts of the plane. Therefore, no camera, regardless
732 of how much time we spend in computing it, can frame them
733 together. In the flight assistants scenario, for example, this hap-
734 pens because one of the flight assistants stays in the back of the

735 plane for most of the simulation, while the other stay located in
736 the front part of the plane. In the other scenarios, passengers
737 exit at the same time from doors that are located at both sides
738 of the plane, and, especially when target recognizability is set,
739 only one door can be framed at each time. When event cover-
740 age is selected, instead, the system manages to find cameras
741 that frame passengers at longer distances, exiting from different
742 doors (e.g. all doors on the same side of the plane).

743 From this and other experimental activity we performed, we
744 can draw some indications on how to set the time available to
745 the *Camera Operator* module. First, one should choose a time
746 that is compatible with a target frame rate. Then, one can mul-
747 tiply the chosen time by a number of frames, so that the total
748 computation time will guarantee good results in the scenario at
749 hand. More specifically, the total time is mostly a function of
750 the maximum number of targets that needs to be framed at the
751 same time (in our examples, a time budget of 21 milliseconds
752 was enough for 10-12 targets). Finally, the number of consec-
753 utive frames over which a camera computation is carried out
754 should be limited since delaying too much the presentation of
755 the newly found camera might cause some brief events to be
756 missed.

757 All data were obtained on a 2.3 GHz Intel Core i7 proces-
758 sor with 16 GB RAM and an NVidia GeForce FT 750 M. With
759 this machine, the simulation, including the camera control sys-
760 tem, runs at between 30 and 60 fps, depending on the number
761 of animated characters displayed, with an average of 40 fps.
762 As explained above, the frame rate is largely dictated by the
763 simulation, as the cost of the camera control system is at most
764 7 milliseconds per frame for a few consecutive frames for the
765 *Camera Operator* module, plus the operations of the *Director*
766 module, which cost about 1 millisecond and is executed every
767 0.2 seconds. Figure 7 shows the milliseconds spent by simula-
768 tion, rendering and camera control code executed on the CPU
769 over a period of 300 frames in one of the exit scenarios pre-
770 sented above. Green bars refer to rendering preparation calls,
771 which take the majority of time because of the large number
772 of animated characters. The bright orange and bright cyan bars
773 refer to the *Camera Operator* module, which is executed every
774 few seconds, with a maximum cost of around 7 milliseconds per
775 frame (the two bars respectively refer to the cost of the called
776 method, and the cost of the subcalls). The zoomed image in the
777 top right part of the Figure shows instead, in red, the cost of the
778 *Director* module, which is practically negligible compared to
779 rendering preparation.

780 5. Conclusions and Future Work

781 In this paper, we have presented a novel application of au-
782 tomatic camera control to emergency simulations and demon-
783 strated the system on a detailed aviation case study.

784 Our system extends a recent virtual camera computation ap-
785 proach to extract interesting events from a simulation, solve vir-
786 tual camera computation problems, analyze the results, and de-
787 termine the virtual camera used to present events of interest to
788 a viewer. As shown in Section 4, our method allows us to visu-
789 alize different aspects of aircraft evacuation scenarios without

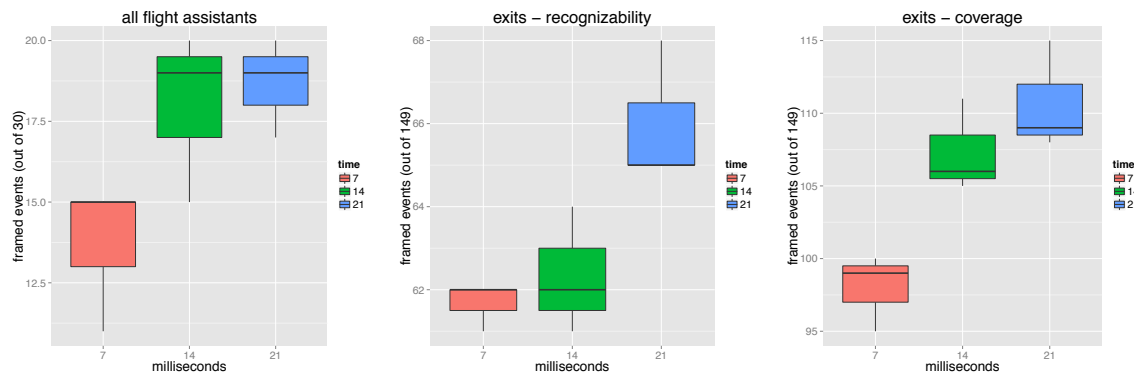


Figure 6: Distribution of the number of framed events in the considered scenarios, with three different time budgets for computing cameras: 7 ms in one frame, 14 ms in two consecutive frames (7 per frame), and 21 ms in three consecutive frames (7 per frame). Data obtained with 50 runs per condition.

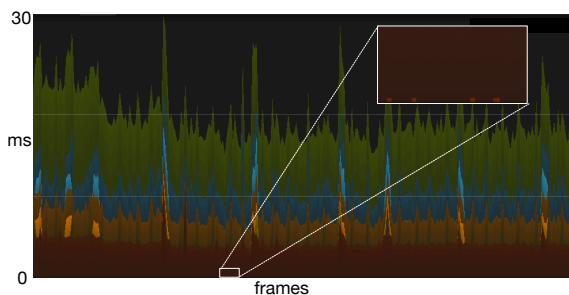


Figure 7: Milliseconds spent by simulation, rendering and camera control code in one of our scenarios over a period of 300 frames. Green bars refer to rendering calls; bright orange and cyan bars refer to the *Camera Operator* module. Bright red bars (zoomed in in the top-right of the image) refer to the *Director* module.

any camera modeling, programming, or control effort by the user. The system is not aviation specific, and could be applied to other safety domains. In particular, one of our next research directions is to apply the system to fire emergencies in buildings. Another interesting potential field of application is video games testing, particularly multi-player games, and perhaps in addition to automatically collected metrics [30]. To this end, it would be interesting to improve event filtering such that more sophisticated rules can be expressed, for example, based on the temporal relations between events.

We plan to conduct a formal evaluation of the system with aviation professionals. However, carrying out a formal experiment in which the system is contrasted to a control condition (simulator without automatic camera control) would probably create an unfair comparison. It would indeed require the user to take charge of camera control in the non-automated condition and, from our own experience, the workload that manual camera control generates makes it difficult to follow the events with the same ease as automatic camera control. Moreover, when several events happen very closely in time, it can be even impossible for the user to manually follow them.

We also plan to improve the camera control method. A straightforward extension would be the possibility of simultaneously computing and visualizing more cameras when one camera is not sufficient to cover current events. This could be implemented by simply requesting the *Camera Operator* module to immediately compute additional virtual cameras when the current virtual camera is missing certain targets and by setting such targets as the ones to be framed. A more general solution would be to change the virtual camera computation algorithm such that the algorithm is able to return multiple solutions instead of only one, i.e., considering the virtual camera computation problem as multi-objective optimization. However, to the best of our knowledge, no camera control approaches with such capabilities have been developed.

A final interesting issue is the addition of high-level knowledge in the virtual camera computation and selection process. Currently, the system reacts to events that are occurring at the moment of changing the current camera without attempting to establish a correlation between past and present events based on their meaning. An ideal visualization of an emergency simulation should instead be able to derive causal relationships between events and perform virtual cameras computation and editing such that these relationships are effectively conveyed to the viewer.

References

- [1] Cha M, Han S, Lee J, Choi B. A virtual reality based fire training simulator integrated with fire dynamics data. *Fire Safety Journal* 2012;50:12–24.
- [2] Mól ACA, Jorge CAF, Couto PM. Using a Game Engine for VR Simulations in Evacuation Planning. *IEEE Computer Graphics and Applications* 2008;28(3):6–12.
- [3] Tsai J, Fridman N, Bowring E, Brown M, Epstein S, Kaminka G, et al. ESCAPES: evacuation simulation with children, authorities, parents, emotions, and social comparison. In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems; 2011, p. 457–64.
- [4] Ranon R, Urli T. Improving the efficiency of viewpoint composition. *IEEE Transactions on Visualization and Computer Graphics* 2014;20(5):795–807.

- 850 [5] Gwynne S, Galea ER, Owen M, Lawrence PJ, Filippidis L. A review of
851 the methodologies used in the computer simulation of evacuation from
852 the built environment. *Building and Environment* 1999;34(6):741–9.
- 853 [6] Owen M, Galea ER, Lawrence PJ. The Exodus Evacuation Model Applied
854 To Building Evacuation Scenarios. *Journal of Fire Protection Engineering*
855 1996;8(2):65–84.
- 856 [7] Galea E, Perez Galparsoro J. A computer-based simulation model for
857 the prediction of evacuation from mass-transport vehicles. *Fire Safety*
858 *Journal* 1994;22(4):341–66.
- 859 [8] Galea ER, Blake SJ, Lawrence PJ. The airEXODUS evacuation model
860 and its application to aircraft safety. In: *International Aircraft Fire and*
861 *Cabin Safety Research Conference*. 2001.
- 862 [9] Johnson CW. Using evacuation simulations for contingency planning to
863 enhance the security and safety of the 2012 olympic venues. *Safety Science*
864 2008;46(2):302–22.
- 865 [10] Chittaro L, Ranon R. Web3D technologies in learning, education and
866 training: Motivations, issues, opportunities. *Computers & Education*
867 2007;49(1):3–18.
- 868 [11] Guttormsen Schär S, Krueger H. Using new learning technologies with
869 multimedia. *IEEE MultiMedia* 2000;7(3):40–51.
- 870 [12] Xu Z, Lu X, Guan H, Chen C, a.Z. Ren . A virtual reality based fire
871 training simulator with smoke hazard assessment capacity. *Advances in*
872 *Engineering Software* 2014;68:1–8.
- 873 [13] Smith SP, Trenholme D. Rapid prototyping a virtual fire drill environment
874 using computer game technology. *Fire Safety Journal* 2009;44(4):559–
875 69.
- 876 [14] Johnson CW. Applying the lessons of the attack on the world trade center,
877 11th September 2001, to the design and use of interactive evacuation sim-
878 ulations. In: *Proceedings of the SIGCHI conference on Human factors in*
879 *computing systems - CHI '05*. New York, New York, USA: ACM Press;
880 2005, p. 651–60.
- 881 [15] Olivier P, Halper N, Pickering JH, Luna P. Visual Composition as Opti-
882 misation. In: *Artificial Intelligence and Simulation of Behaviour*. 1999,
883 p. 22–30.
- 884 [16] Bares WH, McDermott S, Boudreaux C, Thainimit S. Virtual 3D camera
885 composition from frame constraints. In: *Proceedings of the eighth ACM*
886 *international conference on Multimedia*. ACM Press; 2000, p. 177–86.
- 887 [17] Li TY, Cheng CC. Real-time camera planning for navigation in virtual
888 environments. In: Butz A, Fisher B, Krger A, Olivier P, Christie M,
889 editors. *Smart Graphics*; vol. 5166 of *Lecture Notes in Computer Science*.
890 Springer Berlin Heidelberg. ISBN 978-3-540-85410-4; 2008, p. 118–29.
- 891 [18] Oskam T, Sumner RW, Thuerney N, Gross M. Visibility transi-
892 tion planning for dynamic camera control. In: *2009 ACM SIG-*
893 *GRAPH/Eurographics Symposium on Computer Animation - SCA '09*;
894 vol. 1. New York, New York, USA: ACM Press; 2009, p. 55–65.
- 895 [19] Christie M, Normand J, Olivier P. Occlusion-free Camera Control for
896 Multiple Targets. In: *Proceedings of the ACM SIGGRAPH/Eurographics*
897 *Symposium on Computer Animation*. Eurographics Association; 2012, p.
898 59–64.
- 899 [20] Halper N, Helbing R, Strothotte T. A Camera Engine for Computer
900 Games: Managing the Trade-Off Between Constraint Satisfaction and
901 Frame Coherence. *Computer Graphics Forum* 2001;20(3):174–83.
- 902 [21] Lino C, Christie M, Lamarche F, Schofield G, Olivier P. A Real-time
903 Cinematography System for Interactive 3D Environments. In: *2010 ACM*
904 *SIGGRAPH / Eurographics Symposium on Computer Animation*. 2010,
905 p. 139–48.
- 906 [22] Hornung A, Lakemeyer G, Trogemann G. An Autonomous Real-Time
907 Camera Agent for Interactive Narratives and Games. In: *Proceedings of*
908 *the IVA 2003: 4th International Working Conference on Virtual Agents*.
909 Isee, Germany: Springer-Verlag; 2003, p. 236–.
- 910 [23] He Lw, Cohen MF, Salesin DH. The virtual cinematographer: a paradigm
911 for automatic real-time camera control and directing. In: *SIGGRAPH*
912 *'96: Proceedings of the 23rd annual conference on Computer graphics*
913 *and interactive techniques*. ACM Press; 1996, p. 217–24.
- 914 [24] Amerson D, Kime S, Young RM. Real-time cinematic camera control for
915 interactive narratives. In: *Proceedings of the 2005 ACM SIGCHI Inter-*
916 *national Conference on Advances in computer entertainment technology*.
917 Valencia, Spain: ACM Press; 2005, p. 369–.
- 918 [25] Galvane Q, Christie M, Ronfard R, Lim CK, Cani MP. Steering Behaviors
919 for Autonomous Cameras. In: *Proceedings of the Motion on Games -*
920 *MIG '13*. New York, New York, USA: ACM Press; 2013, p. 71–80.
- 921 [26] Eberhart RC, Kennedy J. Particle swarm optimization. In: *Proceedings*
922 *of the IEEE International Conference on Neural Networks* 1995; vol. 4.
923 1995, p. 1942–8.
- 924 [27] Airbus . A320 Airplane Characteristics for Airport Planning, Tech-
925 nical Manual. 2014. [Http://www.airbus.com/support/maintenance-](http://www.airbus.com/support/maintenance-engineering/technical-data/aircraft-characteristics/)
926 [engineering/technical-data/aircraft-characteristics/](http://www.airbus.com/support/maintenance-engineering/technical-data/aircraft-characteristics/) (last access on 25 July
927 2014).
- 928 [28] National Transportation Safety Board . Loss of Thrust in Both Engines
929 After Encountering a Flock of Birds and Subsequent Ditching on the
930 Hudson River, US Airways Flight 1549, Airbus A320-214, N106US,
931 Weehawken, New Jersey, January 15, 2009. Aircraft Accident Report
932 NTSB/AAR-10 /03. Tech. Rep.; National Transportation Safety Board;
933 Washington, DC, USA; 2010.
- 934 [29] Unity Technologies . Unity 4. 2014. [Http://unity3d.com/](http://unity3d.com/) (last accessed
935 on 25 July 2014).
- 936 [30] Moura D, el Nasr MS, Shaw CD. Visualizing and understanding players’
937 behavior in video games: Discovering patterns and supporting aggrega-
938 tion and comparison. In: *ACM SIGGRAPH 2011 Game Papers*. SIG-
939 GRAPH '11; New York, NY, USA: ACM. ISBN 978-1-4503-0970-7;
940 2011, p. 1–6.