ROK KRULEC, B. Eng.
MILAN BATISTA, D. Sc.
Univerza v Ljubljani
Fakulteta za pomorstvo in promet
Pot pomorščakov 4, SI- 6320 Portorož, Republika Slovenija
E-mail: Rok.Krulec@fpp.edu
E-mail: Milan.Batista@fpp.edu

Science in Traffic
Review
U. D. C.: 656.25:656.084
Accepted: Nov. 27, 2002
Approved: Sep. 30, 2003

USER INTERFACE FOR THE SMAC TRAFFIC ACCIDENT RECONSTRUCTION PROGRAM

ABSTRACT

This paper describes the development of the user interface for the traffic accident reconstruction program SMAC. Three basic modules of software will be presented. Initial parameters input and visualization, using graphics library for simulation of 3D space, which form a graphical user interface, will be explained in more detail. The modules have been developed using different technologies and programming approaches to increase flexibility in further development and to take maximum advantage of the currently accessible computer hardware, so that module to module communication is also mentioned.

KEYWORDS

traffic accident reconstruction, Simulation Model of Automobile Collisions, SMAC, user interface

1. INTRODUCTION

It is well known that from the theoretical point of view there are two basic methods for collision modelling used in accident reconstruction programs. The first is the so called impulse-momentum method. The origin of this method is the classical Poisson impact theory [8] and was developed for accident reconstruction in the sixties [2]. The assumption of this method is that during impact the vehicles do not move and that the influence of all forces except impulse is negligible. There are several computer programs which implement this method and in all of them the user must in some way specify the impact point and the impulse direction [9], [11]. These data, which are responsible for post-collision motion, can only be guessed, so that the objectivity of an accident reconstruction is impaired.

The second method treats a vehicle as a deformable body, so that collision forces and moments, and in addition the damage profile can be computed from vehicle crush properties. This approach was introduced by the SMAC program, whose main goal was to improve the objectivity of highway accident investigations [3], [6]. From the estimated initial position and velocities, tyre/road friction data, vehicle data, and driver control data (braking, steering), SMAC calculates vehicle trajectories and damage profiles. From several runs the user can produce the optimal match between the actual accident skid marks and the vehicle damage profiles.

SMAC stands for Simulation Model of Automobile Collisions. The program was developed in the seventies in the US by CALSPAN (Cornell Aeronautical Laboratory) under contract to the US National Highway Traffic Safety Administration (NHTSA). The first version of the program appeared in 1974 and NHTSA sponsored work was done in 1979. Several corrections, improvements and extensions of the SMAC were implemented during the last two decades by program creators and others. These include, for example, impulsive constraints, vehicle-trailer options, tyre deflation simulation, more than two terrain zones and collision logic [1], [4], [5], [7], [8]. Today, the program creator offers the improved SMAC program by the name m-SMAC; other commercial versions are EDSMAC and WinSMAC.

Since the SMAC computer program is one of the most regularly used programs for vehicle accident simulation we decided to develop the Windows-based GUI for SMAC program with full accident animation facilities. In the first stage we developed a program called HiSAC which implemented among others:

- user-friendly menu-driven pre-processors to produce a fixed format SMAC input file,
- graphics post-processor to present results of calculation as diagrams.

In the second stage we will try to implement all the known program extensions and at the same time develop the Windows-based GUI with full accident animation facilities. This paper reports on the current state of the above tasks.

2. HISMAC SOFTWARE PACKAGE

An accident reconstruction software package should allow simple input of initial parameters, fast

calculation and real-time visualization to allow user iteration through accident reconstruction, changing initial parameters to achieve final results that would best match the real accident scene.

HiSMAC is written for PC computers with the Windows operating system, as this is the most widespread computer platform which has enough processing power for fast calculation and good quality visualization.

HiSMAC consists of three main modules:

1. input (HiSMAC Input)
2. calculation (SMAC program) and
3. visualization (HiSMAC 3D)

Modules have been developed using different technologies and programming approaches to increase flexibility in further development and to take maximum advantage of the currently available computer hardware.

HiSMAC Input is written in Object Pascal programming language and compiled with Delphi 5 software development suite. Frame-based and component-based visual development is a powerful tool for developing large scale applications without losing code structure and readability, so that the future enhancements are cleanly implemented, extending the life-cycle of the program.

The SMAC program was written in 1974 in FORTRAN IV programming language and in this form compiled cleanly on 2002 version of Compaq Visual Fortran software development suite. This proves that FORTRAN is the best solution for long-term portability, which is a good choice for a mathematical model which is independent of trends and underlying hardware and software.

HiSMAC 3D was written in Object Pascal programming language, using Microsoft DirectX 9.0b Retained Mode libraries for 3D graphics. DirectX is currently the best choice for visualization on the Windows operating system because it offers up-to-date support for all the graphic processors and technologies which are emerging at a fast rate. A multilayer graphics programming approach has been used to separate 3D simulation graphics programming from the underlying libraries to extend the life-cycle of HiSMAC 3D.

2.1 Input

HiSMAC Input is the standalone Windows program, which is split in several logical units/frames to provide an intuitive input of initial event parameters for calculation and visualization. The following frames are available:

- event,
- vehicles,
- scenery,
- calculation,
- visualization.

The event frame is used to input parameters regarding an accident, such as event description, timing and stop conditions. The user can set start and stop time of simulation, which tells the calculation module to output the calculated data only in the interval from start time to stop time. If start time is set to more than zero, calculation begins at zero, but output data are written only from the start time onwards so that visualization, which relies on the calculated output data is possible only for the set time interval. The user can declare the minimum velocity of the fastest vehicle or the minimum angular velocity of any vehicle to terminate the simulation.

The vehicle's frame is used to describe each vehicle's properties and initial conditions in the event. Each vehicle has its own ID, describing the model and make of the vehicle which is used as reference to pass the vehicle data from HiSMAC Input to SMAC and HiSMAC 3D visualization module. The vehicle properties alone can be saved to a file and then loaded when needed. Initial vehicle conditions describe the start position of each vehicle, its yaw angle and yaw rate at the beginning of the trajectory. Longitudinal and lateral velocity components are set separately, because it is not necessary for the velocity vector to have the same angle as yaw angle of the vehicle (e.g. the case of a sliding vehicle). Vehicle dimensions and location of axles and centre of gravity are exactly specified by setting: the distance of the centre of gravity to both axles and front, rear and side ends and mean track width for front and rear wheels. Rear axle angle, if damaged, is also specified. Yaw inertia property is automatically calculated by [7]

$$I = m \cdot \frac{length^2 + width^2}{12} \tag{1}$$

when any value that has an influence on yaw inertia changes. Yaw inertia can be set to a desired value, regardless of the related values, if necessary. One of the values that affect the value of yaw inertia is also the mass of the vehicle that can be set manually or loaded from file with other car properties.

Cornering stiffness of every tyre can be set individually to allow simulation of broken tyres. Since the values of tyre cornering stiffness are hard to obtain, the HiSMAC uses the following empirical model for calculation of a normal tyre's stiffness [7]:

$$C_{\alpha,0} = -42.06 F_z [N/rad] \tag{2}$$

where is static vertical load on a wheel. Tyre stiffness is also greatly influenced by inflation pressure so that HiSMAC simulates the effect of inflation by the simple linear model:

$$C_\alpha = C_{\alpha,0}(0.2 + 0.8p) \tag{3}$$

where $0 \leq p \leq 1$. Note that in this model 20% of normal stiffness means a flat tyre.

The scenery frame contains controls for input of parameters that describe 3D space and vehicle behaviour on different surfaces. Terrain boundaries are specified with two points which separate two zones with different friction coefficients. The friction coefficient for each zone at zero speed is input together with the coefficient of linear decrement of friction with tyre speed. For describing and positioning of objects in 3D scene, a memo field containing a short script is used. Object-oriented scripting language used is called HiSMAC Script. The example script follows:

Crossroad object is loaded from crossroad.msh mesh file, which is in Microsoft DirectX format, positioned at specified coordinates under specified angle and scaled.

```
sys.load(data\crossroad.msh;0;-2.5;0;0;50)
```

Vehicle polo-red is defined, scaled and coloured with RGBA color value and linked to a file with vehicle movement data.

```
sys.vehicle(polo-red;1.1;11862016;veh1.dat)
```

Another vehicle is defined.

```
sys.vehicle(polo-blue;1.1;181;veh2.dat)
```

Each vehicle loads movement data from the corresponding data file and prepares for simulation.

```
polo-red.simulate()
polo-blue.simulate()
```

Initial camera position in 3D space is set.

```
sys.camera_pos(150;100;40)
```

Defines the object for camera to follow.

```
sys.camera_lookat(polo-blue)
```

Ambient light can be also set.

```
sys.ambient_light(100)
```

These commands are only to illustrate the capabilities of 3D simulation engine as HiSMAC_Script contains even more commands to fine-tune the appearance and positioning in 3D scenery.

The calculation frame contains calculation constants which directly influence the behaviour of external SMAC calculation module, for example interval between radial vectors indicates how many vectors should be used to calculate the damage on the crush surface.

Visualization option runs the HiSMAC 3D visualization module.

There are some parameters which cannot be set yet by the HiSMAC Input, but are also used by the SMAC program in calculation. These must be set directly in an INPUT.DAT file and contain steering and braking information for each given time interval.

Properties of all the above mentioned frames can be saved to a project file and loaded when needed. Before visualization, they are written to INPUT.DAT file which is then used by the SMAC module. 3D scenery data are written to CURRENT.HSC file which is later used by HISMAC 3D visualization module.

Input parameters mostly have measurement units, but the user should decide which standard or magnification to use. To simplify the input of parameters TFlexEdit control has been created to allow real-time conversion between common units and allow mathematical expressions to be evaluated in any field. TFlexEdit control knows types of measures like length, angle, time, velocity, torque, moment of inertia, mass, force, pressure and some others and can convert between various units of the same type. A useful feature of TFlexEdit control is an expression evaluator that can evaluate arithmetic operators (addition, subtraction, division, multiplication, power, div for integer division, mod for residual of integer division, brackets for concatenating expressions and unary + and - signs) as well as Boolean (not, and, or, xor), bitwise (shl, shr, and, or, xor, not), some high level mathematical functions (trunc() for rounding numbers towards zero, round() for rounding numbers towards the nearest whole number and a lot of others), constants (pi, e) and even if() condition sentence as seen in Microsoft Excel. TFlexEdit also stores the given value to a matrix that is later written to INPUT.DAT file, to be used by SMAC module.

2.2. Calculation

Calculation in HiSMAC is entirely based on the SMAC program developed in the 1970's and only the output has been changed slightly to provide all the necessary data. In the future, the program structure will be changed to improve readability and manageability of the source code and allow further development.

There are different types of mathematical models for accident reconstructions differing in the direction of calculation of the accident and phases of the accident included. The SMAC model is an open-form model, which means that the user must specify the initial estimate of the impact speed which generally requires iterations to achieve an acceptable match of the accident evidence. SMAC includes all the phases of the accident:

- initial trajectory phase until impact,
- impact compression phase,
- impact restitution phase,
- after-impact trajectory phase until stopped.

The SMAC model in its current state only works with one or two non-articulated vehicles involved in an accident; further versions will improve the functionality.

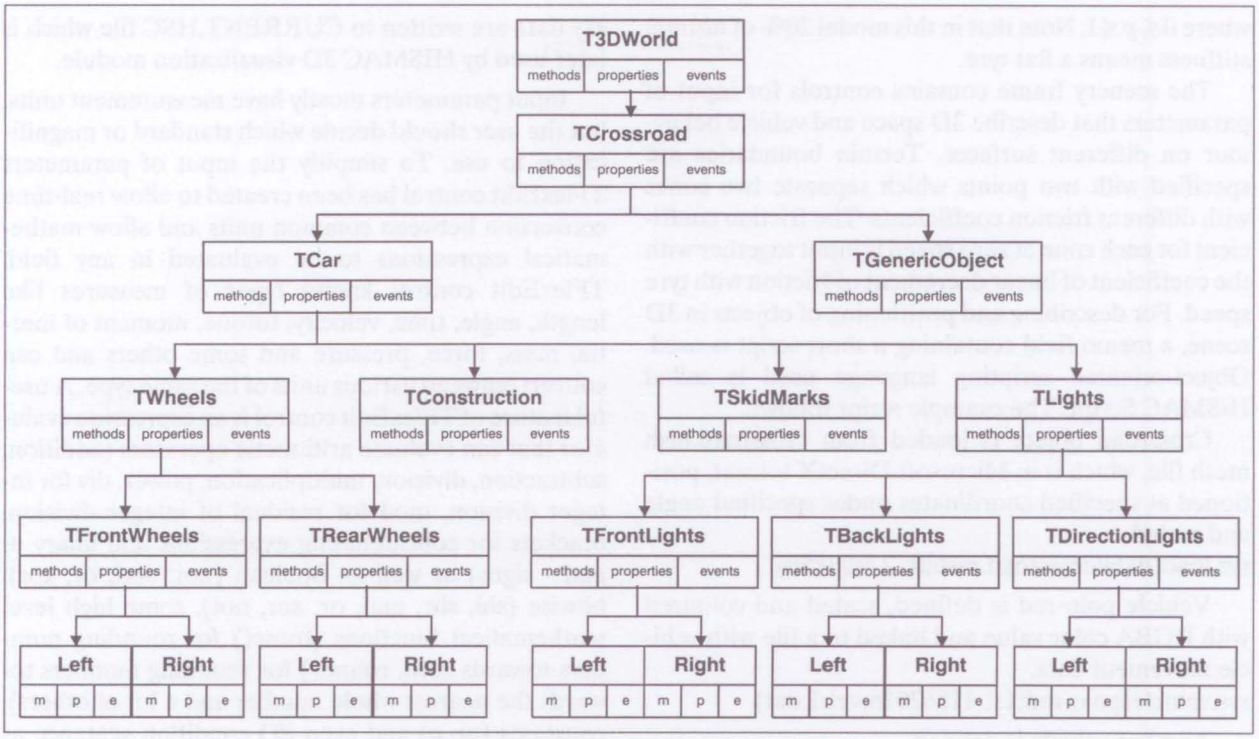

Figure 1 - Object hierarchy of T3DWorld

2.3. Visualization

HiSMAC 3D module used for visualization is a Windows program, using Microsoft DirectX 3D graphics library for rendering 3D graphics. A multi-layer graphics programming approach has been used to separate 3D simulation graphics programming from underlying libraries to extend the life-cycle of HiSMAC 3D. Current layers are:

1. HAL - Hardware abstraction layer;
2. Low level 3D engine - DirectX Immediate Mode;
3. High level 3D engine - DirectX Retained Mode;
4. 3D Simulation engine T3DWorld written in DirectX Retained Mode.

The first layer is written by a graphics hardware manufacturer and is better known as a graphics card driver. The graphics card must support DirectX and all the functions that are defined by the DirectX specification written by Microsoft. The second layer is DirectX Immediate Mode, which supports direct 3D space matrix handling and some utility routines for matrix math. The second layer uses the first layer to communicate with the device. The third layer is a higher level 3D library which uses the second layer to render 3D graphics. It supports DirectX mesh loading and all transformations based on vector/time definitions. The fourth layer is a 3D Simulation engine T3DWorld which is written in Object Pascal and uses the third layer, DirectX Retained Mode, to render 3D space and to move the simulated 3D objects. This layer is the only one that has to be changed if we decide to switch to some other 3D platform than Microsoft DirectX. The following figure describes the object hierarchy of the fourth layer:

Each object in T3DWorld hierarchy has its own methods, which orders a certain object to do something, properties that can be set and events that are triggered by the object.

Although calculations are made in the 2D plane, 3D visualization can greatly increase the understanding of events by seeing the real angles of visibility during the accident, etc. HiSMAC 3D module offers visualization from any angle of the 3D world at any time, by real-time rendering of the 3D graphics. The user can position the camera to any place with help of a hemispherical camera movement and zoom, select the

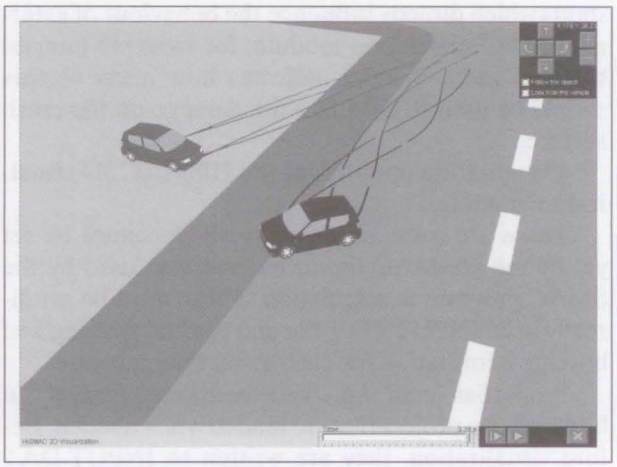

Figure 2 - HiSMAC 3D visualization module

camera look-at object with a mouse or even set the camera to follow the vehicle for a specified distance. The user can also "sit behind the steering wheel" and see the accident from inside the vehicle, which can increase the understanding of the visibility at the time of the accident.

Control buttons are used to play, pause and rewind the accident preview, but with the help of slide bar, the user can go to any time of the accident and see the details step by step in time intervals of 1/100 of a second or more and change camera positions during every step.

2.4. Communication

The main program is HiSMAC Input which sets all the necessary parameters in files INPUT.DAT and CURRENT.HSC to be used by SMAC and HiSMAC 3D modules. Just before visualization, HiSMAC Input saves the parameters in INPUT.DAT and runs HiSMAC Calc to calculate vehicle positions based on parameters in INPUT.DAT. Vehicle positions are then written to file VEHx.DAT. When execution control gets back from SMAC, HiSMAC Input runs the HiSMAC 3D visualization tool which reads vehicle movement data from VEHx.DAT files and scenery data from CURRENT.HSC file. Communication of modules by means of files is a very old approach, but it is the most robust one and can be implemented in a wide variety of programming languages.

Program flow is described by the flowchart in Figure 3.

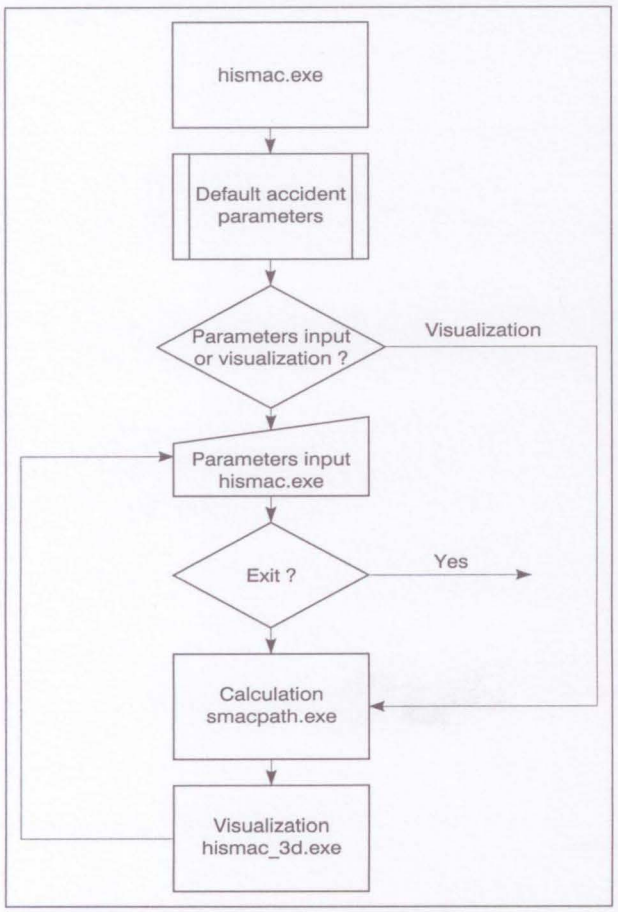

Figure 3 - Program execution flowchart

3 CONCLUSION

An effort is underway to improve the SMAC calculation engine to a more readable form and to allow participation of more than one vehicle. In the long term we would like to add the crush analysis as the base for 3D mesh deformations of the vehicles. 3D engine is being improved on a daily basis with new additions and speed/performance optimizations.

Everyone is invited to HiSMAC official Web site at http://www.hismac.fpp.edu to test a trial Beta version of our accident reconstruction software.

ROK KRULEC, B. Eng.
MILAN BATISTA, D. Sc.
Univerza v Ljubljani
Fakulteta za pomorstvo in promet
Pot pomorščakov 4, SI- 6320 Portorož, Republika Slovenija
E-mail: Rok.Krulec@fpp.edu
E-mail: Milan.Batista@fpp.edu

POVZETEK

UPORABNIŠKI VMESNIK PROGRAMA SMAC ZA REKONSTRUKCIJO PROMETNIH NESREČ

V prispevku je opisan razvoj uporabniškega vmesnika za program za rekonstrukcijo prometnih nesreč SMAC. Opisani bodo trije glavni deli, ki predstavljajo popoln programski paket. Bolj natančno bosta opisana predvsem modul za vnos začetnih parametrov in grafični modul za simulacijo v 3D prostoru, ki sestavljata uporabniški vmesnik. Pri razvoju so uporabljene različne tehnologije in programski pristopi, z namenom povečanja fleksibilnosti programa pri nadaljnem razvoju ter maksimalnega izkoriščanja kapacitet, ki jih ponuja trenutna računalniška strojna oprema.

KLJUČNE BESEDE

rekonstrukcija prometnih nesreč, simulacijski model za trčenja vozil, SMAC, uporabniški vmesnik

LITERATURE

[1] **Terry D. Day**, *An Overview of EDSMAC4 Collision Simulation Model*, SAE Technical Paper 1999-01-0102

[2] **E. Marquard, V. Württ**, *Fortschritte in der Berechnung von Fahrzeug-Zusammenstöen*, ATZ 68/3, pp 75-80

[3] **R. R. McHenry**, *Computer Aids for Accident Investigation*, SAE Paper 760776

[4] **B. G. McHenry, R. R. McHenry**, *SMAC-87*, SAE Technical Paper 880227

[5] **B. G. McHenry, R. R. McHenry**, *SMAC-97 – Refinement of the Collision Algorithm*, SAE Tehnical Paper 970947

[6] **R. R. McHenry**, *Mathematical Reconstruction of Highway Accidents. Scene Measurement and Data Processing System,* NTIS,1975

[7] **R. R. McHenry, B. G. McHenry**, *McHenry Accident Reconstruction 2000*, McHenry Software, 2000

[8] **T. R. Perl, D. O. Anderson, C. Y. Warner**, *Improvements to the SMAC Program,* SAE Tehnical Paper 830610

[9] **H. Steffan, A. Moser**, The Collision and Trajectory Models for PC-CRASH, SAE Paper 960886

[10] **W. J. Stronge**, *Impact Mechanics,* Cambridge University Press, 2000

[11] **R. L. Wooley**, *The IMPACT Computer Program for Accident Reconstruction*, SAE Paper 850245