EXPLAN — a programming language for complex visual stimuli presentation

Romano Colucci\textsuperscript{a}, Carlo Musio\textsuperscript{b}, Cloe Taddei-Ferretti\textsuperscript{a,b}

\textsuperscript{a}Datitalia Processmg S.p.A., Via G. Porzio 9, 80143 Napoli, Italy
\textsuperscript{b}Istituto di Cibernetica del CNR, Via Toiano 6, 80072 Arco Felice, Napoli, Italy

Received 14 April 1993; accepted 12 November 1993

Abstract

Visual cognition research requires the flexible use of structured spatial patterns, characterized by various space and time parameters, which may be administered as visual stimuli. Dealing with this kind of study, we developed a special-purpose programming language and implemented a compiler to build executable programs. The language allows the presentation of stimuli, their space coordinates, persistency values, sequence, kinematic parameters, space–time proximity with other visual stimuli, determination of their modification according to external interaction, generation of subliminal stimuli, monitoring of different subject reactions, automatic reporting of stimulus presentation, and reaction monitoring. Such a language has been successfully utilized in a visual perception research.

Keywords: Programming language; Visual stimulation; Subliminal stimuli; Pattern perception; Visual cognition; Automata theory

1. Introduction

A basic requirement in visual cognition research is the provision of structured spatial patterns to be used as stimuli. A pattern can vary in shape, size, orientation, level of complexity of the contour, internal texture, colour, luminance, and contrast relationship with the background. It could be presented as a single simple figure, or a single figure resulting from the combination of multiple figures, or even as a population of many figures. It could bear no significance, or represent objects and situations, or be ambiguous.

Related to space, a pattern could either be stationary, and then impinge on one of different possible zones of the visual field, or be moving and then be characterized by the type of trajectory, the direction, velocity, and acceleration.

Related to time, a pattern could be either persistent, or appearing and vanishing either slowly or abruptly, or staying during a very short time for tachistoscopic presentation (below the time of a saccade, which is accomplished simultaneously by
both eyes, also if one is covered, and causes a point of the fixated object to drift away from the center of the fovea; saccade duration, 100–300 ms and even less; saccadic movement size, about 10° arc; saccade rate, few minutes of arc per second [1–5]), or during even shorter times for a subliminal stimulation (1–15 ms), or be repeated either alone or in alteration with other patterns according to different time schedules of repetition. It could appear either without conditions, or linked, with variable time relationships, to particular events determined by the experimenter or by the same subject under experiment.

All these needs may be nowadays satisfied in a computer-based setting.

A specifically designed and implemented programming language, named EXPLAN, allows researchers to program complex sequences of visual stimuli and to monitor some significant reactions from the subjects under observation. EXPLAN allows a structural organization to the experiment to be given by organizing the set of visual stimuli involved in the presentation into a finite number of interconnected sequentiation schemes, called the experiment Phases. Addressed functionalities include conditional Phase transitions, periodic sequence generation and event detection.

2. Technical equipment

A personal computer, equipped with a graphics board with a vertical frequency of 70 Hz, a resolution of 320 x 200 pixels and 16 colors, is used to generate the visual patterns. Multiple graphic pages allow quality generation of subliminal stimuli (minimum duration, 14 ms).

We implemented a translator to obtain C language code from EXPLAN symbolic sources and a linkable library to build executable programs.

The host environment was MS-DOS. The translator from EXPLAN to C and the linkable library were written by means of the Microsoft C compiler, which was also needed to compile the generated C code.

An executable program is then obtained by compilation of the generated C code into machine code.

The EXPLAN source generator (EXPGEN) can also be used to obtain powerful and reliable C code control structures, which can be customized to meet more specific requirements.

Image output was obtained by a VGA graphics board, with a resolution of 320 x 200 in 16 gray tones. Subliminal stimuli persistency is 14 ms (and multiples). Valid image file format was uncopressed TIFF, generated by a drafting utility as well as grabbed by an image scanner.

3. Description of the programming language

3.1. What is EXPLAN

EXPLAN is a special purpose computer language. It was designed to control visual stimuli sequentiation in experiments on visual perception. The main supported features are: image flow modification under external interaction, subject stimulation and reaction monitoring, subliminal stimuli generation, automatic report generation of the experiment flow and of the subject reactions.

3.2. Modelling experiments by EXPLAN

3.2.1. Basic concepts

The concepts underlying the EXPLAN language are the following: A number of experiments on visual perception may be carried out by presenting to a subject a set of visual stimuli and recording prominent aspects of the related subject reactions. Each stimulus must have a specific role, which depends on the picture upon which the stimulus is based, its space coordinates, its kinematic characteristics, its persistency on the screen, and the spatial-temporal proximity of other visual stimuli. A stimulus sequence is required, and such a sequence must be variable according to the subject reactions or the interaction of a researcher assisting the experiment.

EXPLAN, apart from implementing such functionalities, allows one to give a time-related structural organization to the set of visual stimuli to be involved in the experiment.

3.2.2. Stimulations

Experiments on visual perception are aimed at measuring a subject’s reactions when stimulated by scenes taken from reality, or by ad hoc gener-
ated patterns. Computers may be used to simulate the presence of such phenomena. In fact, plane pictorial representations may be brought into a computer by means of an image scanner, while three-dimensional world scenes may be captured by means of a camera connected to a frame grabber.

The process of bringing an image into a computer is known as ‘image digitalization’. Digital images are sequences of numbers, so they are stored in the computer as normal data files. The reverse step, i.e. the rendering of stored images, is obtained by means of a graphics board and a color screen.

The computational approach using visual stimuli also includes model simulation in order to investigate various physiological and psychophysical processes occurring in biologic vision [6], such as perception of intensity, color, form and motion, preattentive vision and image recognition [7–10].

3.2.2.1. Basic stimuli. By ‘basic’ stimuli we mean images which must be presented for a time that is at least sufficient to allow full conscious understanding by the subject under observation. We generally talk about basic stimuli when their duration is not under some tens of milliseconds. This is the simplest kind of visual stimulation supported by EXPLAN.

3.2.2.2. Subliminal stimuli. A peculiar feature of the EXPLAN system is the support of subliminal stimuli generation. In fact, a two-page based mechanism allows a complex visual stimulus to be prepared on a hidden page, while normal visualization is performed on the forehead page. At the desired moment, the hidden page can be instantly visualized, and maintained visible for the required time. After that, instantaneous reverse page switching makes the primary page visible again. The minimum allowed time for subliminal visualizations is 14 ms (or multiples thereof).

3.2.2.3. Periodic sequences. A powerful feature of EXPLAN is support for automatic repetition of visual inserts, at fixed intervals of time and with fixed duration. They are called periodic sequences. Multiple periodic sequences are concurrently allowed, even with different periods, giving rise to very complex repeating patterns at a very low programming cost. Precise synchronization among sequences and precise sequence synchronization with reference to a specified event are allowed.

The correct scheduling of the visual stimuli involved in a set of concurrent periodic sequences is managed by a piece of program acting as sequencer.

The sequencer takes into account all stimuli defined to be periodically repeated, continuously watches the computer timer to see if it is time to insert a visual stimulus, alternates ‘cooked’ visual stimuli by means of a private screen page (thus no interference is possible with normal and subliminal program-controlled sequentiations) and reschedules operated stimuli for repetition (Fig. 1).

In relation to the persistency time of auto-repeating stimuli, both the subliminal and the conventional ranges are applicable with no programming difference.

3.2.3. Organizing the stimulation flow: the Phases

As above claimed, EXPLAN allows visual stimuli to be arranged into a control structure, by organizing the set of stimuli involved in the presentation into a finite number of interconnected sequentiation schemes, called the experiment Phases (see Appendix 1). Each Phase, excepted for one special terminal Phase, will bring the image stream control into another Phase, in a deterministic or external intervention dependent fashion, according to the experiment requirements.

We could model the dynamic process governed by an EXPLAN program by means of a finite state automaton. Recalling the concepts of finite automata theory, some (computational) processes may be found only into a finite number of states and the transition from one state into another is due to a particular condition being met. The application of this model to EXPLAN programs is apparent if we parallel automata states with Phases, and state transition conditions with the criterion to pass from one Phase into another.

In the same way each automaton has one initial state, so each EXPLAN experiment must have one Phase declared as initial. Moreover, to be terminated, the experiment must fall into a special, final state, entered by the STOP statement (Fig. 2).

From the programming point of view, Phases should be detected according to the general criterion of obtaining brief, logically unfragmen-
REPEAT Pattern-1 FOR 500 msec EACH 5 sec

As soon as possible

REPEAT Pattern-1 FOR 500 msec EACH 5 sec WAITING 500 msec SINCE Phase_Beginning

First presentation of Pattern-1

REPEAT Pattern-2 FOR 1 sec AFTER EACH Pattern-1 WAITING 3 sec

Fig. 1. Time diagrams for samples of basic operativity with automatically repeating stimulations. Suppose pattern 1 and 2 to be two declared stimuli.
INTENT:
Study on distance perception
Distance variation simulation

STIMULI:
HOT STIMULUS Narrow_Stripes IS NSTRIP.TIF AT 50 50
HOT STIMULUS Large_Stripes IS LSTRIP.TIF AT 50 50
HOT STIMULUS Wide_Stripes IS WSTRIP.TIF AT 50 50

PHASES:
PHASE Instruction
   MESSAGE "Pay attention!" AT 20 10
   WAIT 10 sec
   CLEAR Normal Screen
   PERFORM Distance_Simulation
PHASE Distance_Simulation
   SHOW Narrow_Stripes
   WAIT 500 msec
   SHOW Large_Stripes
   WAIT 500 msec
   SHOW Wide_Stripes
   WAIT 500 msec
   CLEAR Normal Screen
   STOP

FIRST PHASE IS Instruction

---

Fig. 2. Example of an EXPLAN experiment protocol and its translation into the symbolic drawing of automata theory. For details of the grammatical rules see text and Appendix 1.

Table program modules, giving rise to 'visual concepts'. It is somewhat like organizing writing into sentences. More specifically, a first key to structure is the detection of repeating sub-sequences of visual patterns. A second key might be the detection of logically related and temporarily connected visual stimuli.

EXPLAN Phases are made up of three building blocks, among which only the first one is mandatory: the body, the set of Interludes, the set of Gates (see Appendix 1).

The Phase body is the place where the standard set of actions to be carried out by a Phase must be expressed.

Interludes represent a way to insert a variant inside the normal execution of a Phase body. In fact,
INTENT:
An upside-down, small shape is displayed. The subject shall be instructed to press the space bar to zoom the image for a while, and to press either mouse button when he recognises a familiar shape.

STIMULI:
HOT STIMULUS Small Shape IS SM_SHAPE.TIF AT 150 80
HOT STIMULUS Big Shape IS BG SHAPE AT 100 40

PHASES:
PHASE Presentation TIMEOUT 40 sec
SHOW Small Shape
INTERLUDE Space ACTIVE SINCE Phase_Begning
SHOW Big Shape
WAIT 2 sec
CLEAR Normal Screen
SHOW Small Shape
GATE LButton ACTIVE SINCE Phase_Begning
STOP
GATE RButton ACTIVE SINCE Phase_Begning
STOP
FIRST PHASE IS Presentation

Fig. 3. The same as Fig. 2 with different experiment flow specifications. The automata theory drawing is represented without (left) and with (right) Interlude and Gate instructions. For details of the grammatical rules see text and Appendix 1.
an Interlude is a list of actions to be undertaken as a consequence of an external interaction, if it is detected while running a Phase body.

The Phase body is not abandoned as a consequence of an Interlude activation: in fact, after the complete execution of the actions listed in the Interlude, the control goes to the Phase body action following the one last completed before the Interlude activation.

Gates are similar to interludes, in the fact that they list the set of actions to be temporary carried out as a consequence of an external interaction, but they have the fundamental role of allowing transition from one Phase to another. In fact, after the complete execution of the tasks listed in the Gate body, control is not to be returned inside the Phase body. To this aim, the last action in a Gate body must necessarily be a command to pass to another Phase.

Extending the automata model of EXPLAN processes to view the Phases in terms of their body and of the body of interludes and Gates, we must consider each Phase body, each Phase Interlude and each Gate body as a state; arcs stemming from Phase bodics may enter interludes and Gate bodies, or may be representative of the deterministic transition into another Phase; arcs coming from interludes may only reenter the owner Phase; finally, arcs coming from Gate bodies represent the deterministic transition into their consequent Phases (Fig. 3).

3.2.4. Relating activities to time flow: the Events

A Gate or an Interlude associated with a Phase does not have to be necessarily enabled from the beginning of the Phase. In fact, in some cases it may be desirable that at least a set of actions be performed before the standard actions flow listed in the Phase body can be suspended or interrupted.

To this aim, EXPLAN allows one to specify the instant when a Gate or an Interlude starts being active, by referencing either user-defined or predefined time markers, called Events (see Appendix 1). The predefined Events are relative to the experiment beginning, the current Phase beginning, the current (if any) Interlude beginning, or the current (if any) Gate beginning.

3.2.5. Detectable Interactions

EXPLAN supports the detection and reporting of external interactions, for purposes other than flow control modification obtained by Interludes and Gates. Such interactions do not bias the execution of the experiment, but allow some manifest reactions of the subject to be record extemporarily. They are called Detectable Interactions (see Appendix 1).

In the current EXPLAN implementation, interactions are based on the use of the keyboard and mouse buttons.

3.3. The EXPLAN language

3.3.1. The program structure

Structurally, an EXPLAN program is organized into three declarative divisions.

(i) Intent declaration. This is a header to document the experiment significance. Such a documentation is not syntax restricted, so it will be written in plain text.

(ii) Stimuli declaration. Stimuli are described in terms of the raster images to be employed, the screen location where images must be placed, and a persistency attribute to distinguish subliminal stimuli by normal stimuli. This division has the basic role of associating a 'logical' name to each 'physical' stimulus (see Appendix 1).

(iii) Phase declaration. The stimuli flow control is organized inside this division. All Phases, introduced by a logical name, are described in terms of their invariate sequence of actions, possibly in the terms of their interludes and of their Gates. Both Interludes and Gates are, in turn, described in terms of the event in which they become active, and the invariate, uninterruptible sequence of actions entered through them. The Phase declaration division includes the First Phase declaration, that is, the specification of the Phase to be performed as the experiment start Phase.

3.3.2. The EXPLAN grammar

The grammatical rules to write programs using the EXPLAN language are listed in Appendix 1. The grammar is made up of a set of non-terminal symbols, a set of terminal symbols, and a set of production rules (the technical term 'ter-
minal' used here bears a referential significance and not a temporal one) [11].

Production rules must be applied to non-terminal symbols to progressively turn phrases containing terminal and non-terminal symbols into phrases of the language, i.e. containing terminal symbols only.

A non-terminal category is a symbolically referenced item enclosed within angular brackets, as in (Time Specification).

A terminal symbol is a word or a group of words not enclosed within angular brackets.

All terminal symbols in capital letters appearing in the grammar production rules are keyword in the EXPLAN language, i.e. they are the constant building block of a statement. All other terminal symbols are objects: they are the varying parts of EXPLAN statements.

User defined objects may be specified with no limitations on the use of special (non-alphabetic) or numeric characters. However, we suggest following the criterion we employed for the naming of predefined events: words in small letters, started by capital letter, multiple words joined by underscore.

A production rule is a formalization for expressing the fact that a non-terminal symbol may be substituted by (hence, may produce) other symbols, either constant or replaceable in their turn.

A production rule has the generic form:

\[ (A) := B \]

where \( A \) is a non-terminal symbol, and \( B \) is what can be replaced for each occurrence of \( A \). When multiple production rules have the same left hand portion, this means that the same non-terminal symbol may be replaced in different manners. Moreover, if a production rule contains portions enclosed within square brackets, it means the portions are optional. It is a compact notation to avoid writing too many production rules.

The start symbol is the seed of each program. In fact, it is the non-terminal symbol which is able to generate all the EXPLAN programs.

3.4 Dimensional evaluation of EXPLAN environment and applications

The EXPLAN compiler amounts to 2165 C language statements, corresponding to 38 kb of executable code.

Each experimental protocol gives rise to a number of C statements at a ratio of 4:1 with respect to each statement in the EXPLAN source. Moreover, it requires a fixed library support of about 1400 C statements.

Depending on the number of statements required by the experimental flow, an executable experimental protocol does not have a fixed volume, but, for the purposes of a reference dimension, we observed that an EXPLAN protocol of 500 statements corresponds to a program of 70 Kb.

3.5 Hardware-software limitations and possible improvements

3.5.1 Colours

The currently video card employed by us allows the use of a limited number of colours (16) when employing multiple video pages. Commercially available video cards (such as the Imaging Technology VS 100, which manages 256 colours images and 16 pages) enable the presentation of a wider range of visual stimuli. These cards are needed in all cases in which colour is a discriminant component in visual patterns.

3.5.2 Screen refreshing rate

Related to the limitations due to the currently video card employed by us, it must be pointed out that the 70-Hz rate of screen refreshing, allowing the presentation of subliminal stimuli with minimum duration of 14 ms, does not reach the technology limit for affordable cost hardware. In fact, 100-Hz video cards (and the corresponding necessary screen) are now available. The vertical rate of 100 Hz allows 10-ms subliminal stimuli. Moreover, a higher screen refreshing rate gives more permanent images.

3.5.3 Management of extended memory and hidden pages

EXPLAN executable programs (i.e., the *.EXE files) cannot currently use the whole central memory (RAM) available on the computer, but only 640 Kb, because the current implementation of EXPLAN was developed using version 6.0 of
the Microsoft C Compiler. The updated version of the quoted compiler (or the use of other appropriate development environments) enables access to the whole RAM memory available on the computer.

For this purpose, we are planning a new EXPLAN implementation by means of the Microsoft Visual C++ Prof 3.5. In fact, the employment of a greater amount of RAM memory enables the utilization of more ‘hot’ stimuli, since they are preloaded in main memory. Furthermore, proper use of the hidden page in alternation with the forehead one could also allow the generation of simple animations.

### 3.5.4. Embedding other kinds of stimulation

Visual perception experiments may gain greater significance when more complex visual patterns and/or other kinds of sensory stimulations (sound at first) are involved. Given the emerging availability on the market of computer-driven video source processors, digital sound generators and integrated multimedia devices at an affordable cost, we believe that the EXPLAN environment could meet the needs of broader types of investigations, ranging from the assessment of visual impairment to the evaluation of cognitive capabilities related to attentional states.

### 4. Applications of the EXPLAN language

The EXPLAN language has been already tested during the data collection of a research dealing with multistable visual pattern perception [12,13], whose bistable reversible figure perception is a particularly well-studied case [14–20].

Through seven different complex experimental protocols, each one composed of seven different Phases of pattern presentation, and the recording of the types and temporal occurrence of the perceptual events during the experimental tests, it has been ascertained that preferential choices from among the possible pattern perceptions exist, as well as from among the possible temporal patterning of successive shifts between different perceptual states. It has been also ascertained that it is possible to orient the shifting activity in differentiated ways by means of subliminal stimuli having different structure and different temporal relationships with respect to previous perceptual events.

The flexibility of the EXPLAN language makes it suitable for further different experimental protocols in the same research and for other types of visual cognition research. It should also be suitable for investigations with selected groups of the population, in order to evaluate the relationships between physiological or pathological aging and the decay of visual cognitive capabilities [21].

### Acknowledgements

We are grateful to Dr O. Talamo (Istituto di Cibernetica, CNR, Arco Felice) for many fruitful suggestions concerning the hardware equipment and software application, and to Mr A. Cotugno (Istituto di Cibernetica, CNR, Arco Felice) for his continuous skillful assistance. This work was supported in part by a grant from the CNR Special Project ‘Non Linear Dynamics in Biological Systems’.

### Appendix 1 — Rules of the EXPLAN grammar

**Start Symbol**: `<Experiment>`

**Production rules**:

- `<Experiment>` = `INTENT: <Experiment Description>`
- `<Stimuli>` = `<List of Stimuli>`
- `<Phases>` = `<List of Phases>`
  - FIRST PHASE IS `<Phase Name>`
  - `<List of Stimuli>` = `<Stimulus>`
    - [SUBLIMINAL] STIMULUS `<Stimulus Logical Name>` IS `<Stimulus Physical Name>` AT `<Position>`
  - `<List of Phases>` = `<Phase>`
    - `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<List of Phases>` = `<Phase>`
  - `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
- `<Phase>` = PHASE `<Phase Name>` `<List of Actions>` `<List of Interludes>` `<Gate Timeout>`
<List of Interludes> := 4nterlude> := ACTIVE SINCE <Event>
<Interlude Body> := <List of Actions>

<List of Gates> := <Gate>
<Gate Body> := [GATE Detectable Interaction> ACTIVE SINCE <Event>

<Gate Body> := [<List of Actions>]

<Screen Page> := Normal Screen
<Screen Page> := Hidden Screen

<Color> := Black
<Color> := Gray
<Color> := White

<Abscissa> := a positive integer number, or 0
<Coordinate> := any character string
<Event Name> := any character string
<Integer Number> := a positive integer number
<Integer Specification> := <Integer Number> <Time Unit>
<Phase Name> := any character string
<Stimulus Logical Name> := any character string
<Stimulus Physical Name> := a file name

References


