Paper ID: E2-E-05

PHYSICS ERA: STRUCTURE INTERFACE OF PHYSICS ON COMPUTER

NITISH KUMAR
COMPUTER SCIENCE & ENGINEERING DEPARTMENT
SRI SAI COLLEGE OF ENGINEERING & TECHNOLOGY, BADHANI (PATHANKOT)
Email: niku906099@gmail.com

Abstract:

Educators at Indian university haven not started using high level programming language with FOS to teach computational physics. I am going to describes how graphical visualizations also play an important role, which I illustrate with few examples. An every computer student should understand the concept of trinity and floating point .Here the trinity words means: Mathematics, Physics and Informatics. A concept of trinity gave me idea to work on the principle of Physics (classical and quantum) and computer science together. As we know that Mathematics is the mother of computer science and physics gave a structure of computer which gave a new theory & concept of Informatics. Here I explain with concepts of physics and their relation and with high level programming language in FOS as well as in windows, mac-os. While it is possible to introduce the Quantum computation in a strictly algebraic manner without ever mentioning "real world" things like electrons, particle states or charge density, some basic knowledge about general quantum physics can vastly improve the understanding of why certain quantum algorithms or programming techniques actually work and a good precaution against common misconceptions'.

Categories and subject description: Physics engine and quantum computer in open source technology

Keywords: Open source software, quanta, interface, design and application, high programming language

1. INTRODUCTION

In 1990's it was realized that quantum physics has some spectaculars application in computer science. A new audience consists of board concepts for teaching quantum mechanics whose background and needs a require new style of quantum pedagogy. It occupies a unique position in the history of science. This paper is intended primarily for computer science students who know nothing about quantum theory but like to know the computation of these concepts either with curiosity about new paradigm or a basis for further work in their own area.

There will be slight difference between classical and quantum computation and benefits of quanta[1]? Let see... "What will be happening when there will be no use of discrete value, instead of these there will be something like complex number[2] or only rays of propagation[3]"? It is possible instead that future will bring about classical systems that are merely optimized by quantum devices. But even then, it could still lead to some pretty spectacular results. Imagine artificial intelligence powered in part by subatomic particles, just like human brains.

---

1. In physics, a **quantum** (plural: **quanta**) is the minimum amount of any physical entity involved in an interaction.
2. A **number** that can be expressed in the form a + bi, where a and b are real **numbers** and i is the imaginary unit, that satisfies the equation $x^2 = -1$, that is, $i^2 = -1$.

3. To transmit characteristics from one generation to another. Cause to move in some direction or through a medium, such as a wave.

The paper is organized as follows: Section 2 provides the background of quanta or quantum, through which this paper was got some backbone by briefly explaining postulates of quanta. Section 3 deals with the reliable interface contain briefly how to build the quanta network from ions to photons. Section 4 deals with analysis of open software CalcHEP [1], which diagrammatically explains about a good interface for colliding physics particle [2]. Section 5 deals with design and implementation in c++ under various versions of the UNIX operating system by using standard X graphics library system. Section 6 deals with numerical session and related work. Section 7 followed by conclusion and future scope.

2. BACKGROUND

This section explains fundamental discrete value or formation of qbits [3] of quanta with 4 brief postulate of mechanics which will form the basis of interface.

An important distinction needs to be made between quantum mechanics, quantum physics and quantum computing. Quantum mechanics is a mathematical language, much like calculus. Just as classical physics uses calculus to explain nature, quantum physics uses quantum mechanics to understand or explain the nature. Just as classical computers can be thought of Boolean algebra terms, quanta computers are reasoned about with quanta mechanics. There are four postulates [4] to quanta mechanics, which will form the basis of interface.

POSTULATE 1: A quantum bit

"Associated to any isolated physical system is a complex vector space with inner product (i.e., Hilbert space) known as the state of the space system. The system is

completely described by the state vector, which is a unit vector in the system state space". Let a single qubit with 2D state space. Let $|\theta 0>$ and $|\theta 1>$ be orthogonal basis for the space. Then a qubit $|\Psi>=a|\theta_0>+b|\theta_1>$. In quantum computing, we usually label the basis with some Boolean name but note carefully it is only name. Making more concrete one might imagine that "$|\theta 0>$ is being represented by up-spin value and $|\theta 1>$ is represented by down-spin value. Note that qubit $|\Psi>=a|\theta_0>+b|\theta_1>$ should be in unit vector.

POSTULATE 2: Evolution of quantum system:
"The evolution of closed quantum system is described by a unitary transformation that is the state $|\Psi>$ of the system at time t_1 is related to state of $|\Psi'>$ of the system at time t_2 by unitary operator U which depends only on times t_1 and t_2." i.e., $|\Psi'>=U|\Psi>$. The fact that U cannot depend on $|\Psi>$ and only on t_1 and t_2 is suitable and disappointing fact. We will see later that if U could depend on $|\Psi>$ then quantum computers could easily solve NP complete problems! Conceptually think of U as something you can apply to quantum bit but you cannot conditionally apply it. The transform occurs without any regards to the current state of $|\Psi>$.

POSTULATE 3: Measurement:
"Quantum measurement is described by a collision by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\Psi>$ immediately before the measurement then the probability that result m occurs is given by $p(m)=<\Psi|M_m^\dagger M_m|\Psi>$ and the state is $M_m|\Psi> \Big/ \sqrt{<\Psi|M_m^\dagger M_m|\Psi>}$.
The measurement operators satisfy complete equation.$\sum_m <\Psi|M_m^\dagger M_m|\Psi>=I$. The completeness equation expresses the fact that a probability is 1. $\sum_m p(m) = \sum_m <\Psi|M_m^\dagger M_m|\Psi> \sum_m <\Psi|M_m^\dagger M_m|\Psi>$
As a side note I am forced to wonder if Postulate 3 can be derived from Postulate 2. It seems natural given that measurement in the physical world is just interacting qubit with other qubit.

POSTULATE 4: Multi-qubit system:
"The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Suppose systems 1 through n and system I is in the state $|\Psi_i>$, then the joint state of the total system is $\Psi_1>$ tensor product$\Psi_2>\ldots.\Psi_n>$. Here the tensor product exposes the essence of superposition of qubit".

3. A RELIABLE INTERFACE

"In order to build a quantum network with trapped ions, we need an efficient interface that will allow us to transfer quantum information from ions to photons,". In my interface, author has position two ions between two highly reflective mirrors, which form an optical resonator. I entangle the ions with one another and couple both of them to the resonator." The collective interaction between the particles and the resonator can now be tuned in order to enhance the creation of single photons. "This is known as a super radiant state," In order to demonstrate that the interface is well suited for quantum information processing, the researchers encode a quantum state in the entangled particles and transfer this state onto a single photon. The process of information transfer from the particle to the photon essentially becomes more robust," [5]. As a consequence, the technical requirements for the construction of accurate interfaces may be relaxed.

Here, the emission of a photon is suppressed rather than enhanced. "These states are also interesting because the stored information becomes invisible to the resonator, and in that sense, it's protected," As a result, one can imagine that by switching between sub-and super radiant states, quantum information can be stored in ions and retrieved as photons. In a future quantum computer, such addressable read-write operations [6] may be achieved for a quantum register of trapped ions. I have developed a quantum interface which connects light particles and atoms. The interface is based on an ultra-thin glass fiber and is suitable for the transmission of quantum information.

In that sense, glass fiber networks can be considered as the backbone of the modern communication society. The light that travels through them is not a continuous flow of energy. It rather consists, as was discovered by Albert Einstein, [7] of indivisible energy quanta, or photons. Each photon can then transmit one bit of information, corresponding to a zero or one. In addition to being very efficient, this also opens the route towards entirely new ways of communication because, being quantum objects, photons can exist simultaneously in states, zero and one.

Testing of atoms and lights:

"In my experiment, Author connects two different quantum physical systems," "On the one hand author use fiber-guided light, which is perfect for sending quantum information from A to B and on the other hand, I rely on atoms, which are ideal for storing this information."By trapping atoms at a distance of about 200 nanometers from a glass fiber, which itself only has a diameter of 500 nanometers, a very strong interaction between light and atoms can be implemented. This allows one to exchange quantum information between the two systems. This information exchange is the basis for technologies like quantum cryptography and quantum teleportation [8]. Currently, there are different approaches towards performing quantum mechanical operations and exchanging quantum information between light and matter-based memories.

However, for many of these systems it is challenging to store and to retrieve the information efficiently."Our setup is directly connected to a standard optical glass fiber that is nowadays routinely used for the transmission of data, "It will therefore be easy to integrate our quantum glass fiber cable into existing fiber communication networks.". In the past, the researchers already demonstrated that atoms can be controlled and efficiently coupled to glass fibers. However, so far, the suitability of the fiber-coupled atoms for storing quantum information and for long-distance quantum communication remained an open question. After some time, the quantum information stored in the atoms is lost as it leaks into the environment -- an effect called

"decoherence.".[9]."Using some tricks, I was able to extend the coherence time of the atoms to several milliseconds, in spite of their small distance to the fiber surface, "Light in glass fibers travels about 200 kilometers in one millisecond. As the light carries the quantum information, this defines the separation that could be bridged with such a system via the entanglement of atoms.

Important open source software that is CalcHEP, which is used for high level of energy automation particle [10]. A package for the automatic calculation of elementary particle collisions and decays in the lowest order of perturbation theory [11]. The interactive session of CalcHEP is graphical and menu-driven and guides the user through the calculation by breaking the calculation up into a series of steps. At each step, CalcHEP presents the user with the available options allowing him/her to control the details of their calculation in an intuitive way. Moreover, at each menu, contextual help is available which explains the details of the current choices. The CalcHEP package [12] consists of three parts which perform the symbolic, numerical and batch calculations. The first two parts are written in the C programming language. The symbolic part produces C codes for squared matrix elements which are then used in the numerical calculations. The last part is written in Perl. It calls the routines of the first two parts and collects the results to obtain final cross sections and events for multichannel processes typical of modern collider physics [13].

4. ANALYSIS OF CalcHEP

In this section, author would like to discuss the general elements of the CalcHEP graphical user interface. Among these elements are the on-line help, the menu, messages, the string editor, the table editor, the diagram viewer and the plot viewer. The user can control them using the Arrow keys, the Enter key, the Esc key, the Backspace key, the PgUp/PgDn keys and mouse clicks. Additionally, the user can control whether colors and sounds are used and can choose the font used in the graphical user interface. The user can set all of these preferences by editing the calchep.ini file located in his/her work directory. This file contains three lines, one for each of these settings. The color and sound are set by simply toggling between on and off. The font is set by specifying the full font string of the desired font. Other fonts available on the users system can be obtained by the command xls fonts, however, only non-proportional fonts should be used for CalcHEP.

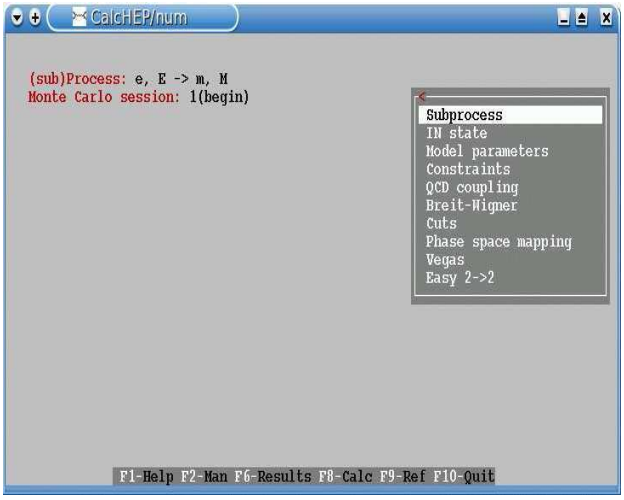

Figure 1. CalcHEP Graphical User Interface

CalcHEP uses a table structure to store information about the model's parameters, particles, and vertices as well as for the cuts and distributions of a numerical calculation. For each of these cases, one row stores the information for one parameter, one particle, one vertex, one cut and one distribution respectively. CalCHEP has a table editor which allows the user to view and, at times, to modify the contents of these tables.

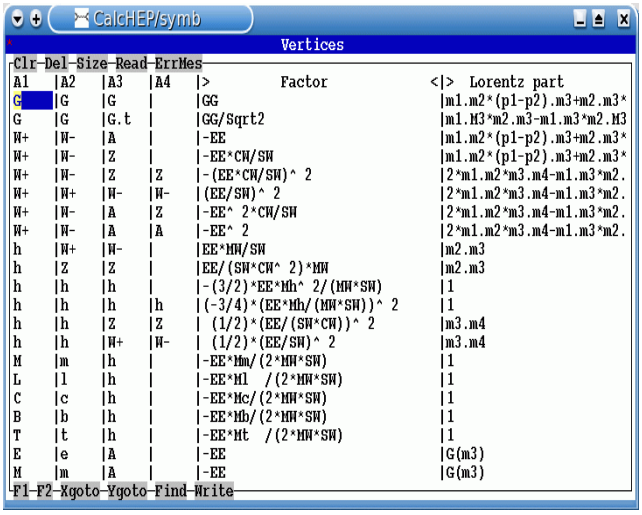

Figure 2. CalcHEP Table Structure

The Diagram Viewer is designed to display multiple Feynman diagrams at a time. The viewer splits the window into rectangular cells and puts one diagram in each cell. Each cell is enclosed by a frame while the current diagram is highlighted by having a thicker frame than the others. The index of the current diagram is displayed at the top-right of the window along with the total number of diagrams.

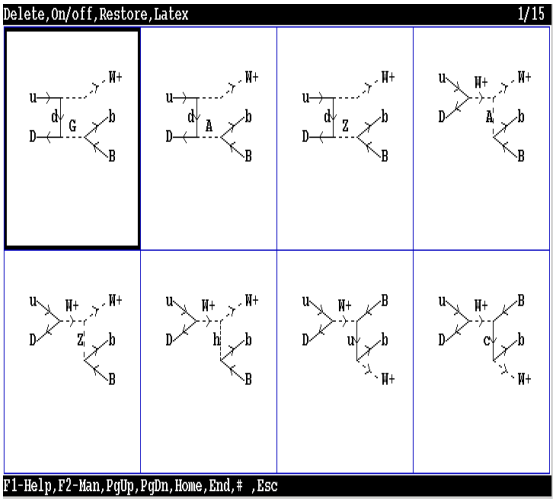

Figure 3. Diagram View

The Plot Viewer is designed to display histograms (see Figure 4) and continuous curves (see Figure 5). After being launched, the Plot Viewer displays the plot and waits for a signal from the keyboard or the mouse. If the mouse is clicked inside the plot, the x-coordinate of the mouse is displayed at the bottom of the window. Additionally, the value of the function or histogram at that x-coordinate value is shown. In the case of a two-dimensional histogram density plot, both the x-coordinate and the y-coordinate along with the histogram height are shown.

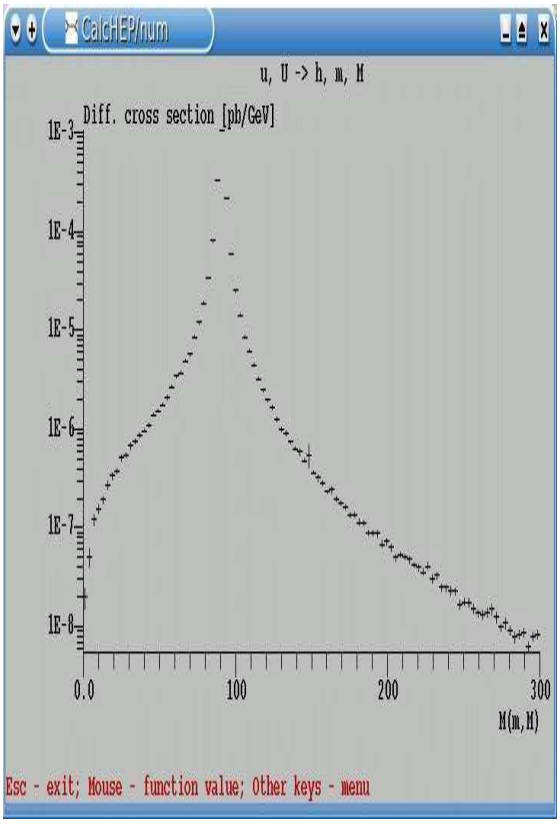

Figure 4. Histogram

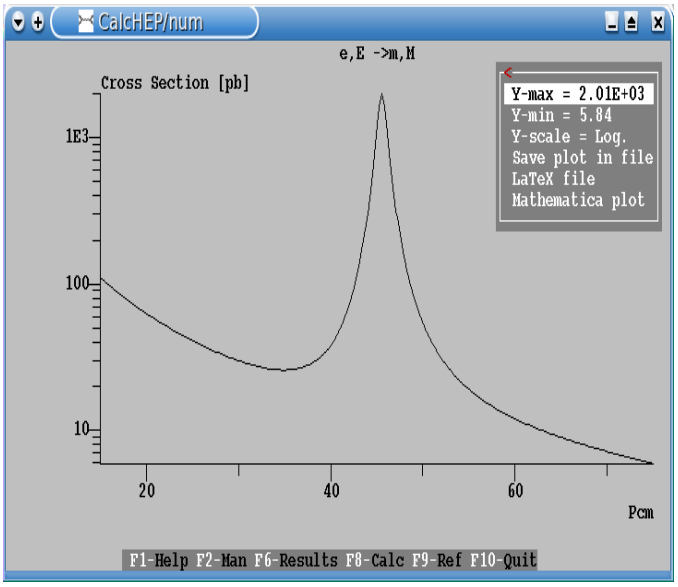

Figure 5. Continuous Curve

Let's move on calculation of decay particle, when calculating the decay widths and branching ratios [14], CalCHEP first calculates the contribution from $1 \rightarrow 2$ decays. If the resulting width is zero, it then calculates the contribution from $1 \rightarrow 3$ decays. If still zero, it calculates the contribution from $1 \rightarrow 4$ decays. However, if the model is defined in terms of an SLHA file and that file contains the widths and branching ratios, CalCHEP does not calculate them, but uses the values specified in the SLHA file.

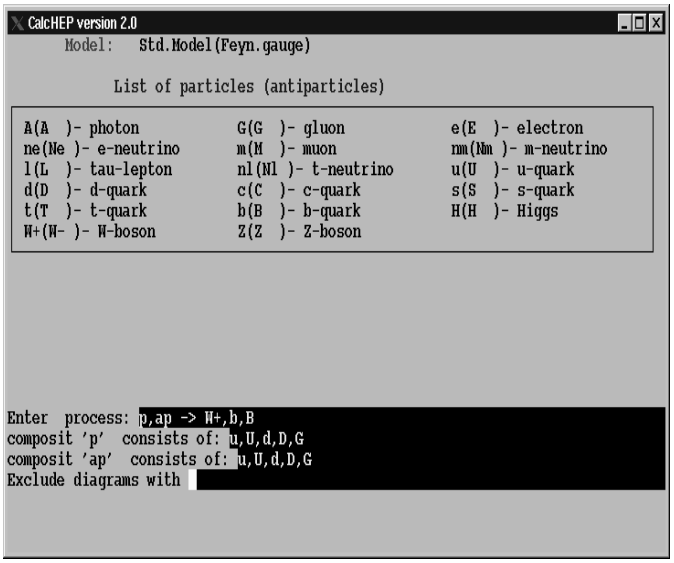

Figure 6. Symbolic Interaction

All menu here has specific task for making a good interface for colliding physics particle. I described the process of generating and compiling C code for a collision or decay process for any particle physics interaction model. In this section, I describe how to use the resulting executable to calculate the collision cross section or the decay width in interactive mode. In Figure 7, author presents a schematic view of the menu system of the interactive numerical session. The first menu item

is Sub process which allows the user to choose which sub process to work on if more than one was generated during the symbolic session. The next menu item is in state which allows entering the momentum of the incoming particles, their polarizations and their Parton distribution functions. The next menu item is Model parameter which allows modifying the numerical value of the independent parameters which are used in the numerical calculations. The Cuts menu item allows setting cuts on the Monte Carlo [15] phase space integration and event generation. The Breit-Wigner [16] menu item allows modifying the behavior of the propagators for the unstable particles. The Phase space mapping menu item opens up into a menu with two items. The Monte-Carlo simulation menu also allows generating kinematic distributions and generating events. For 2->2 processes with fixed energies of incoming particles the phase space integral is one-dimensional and can be integrated using traditional Riemann approach [17]. In this case, one can chose the 1D integration which is the last item in the menu. For the 1-> 2 cases we have zero dimension phase space and this option allows a fast summation over channels and calculation of branching.

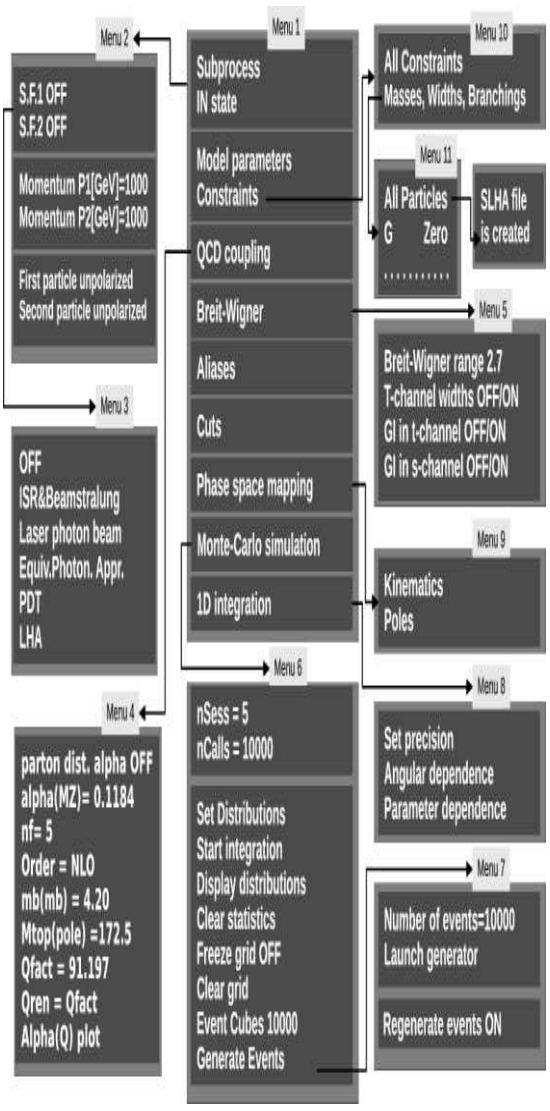

Figure 7. Schematic Diagram of CalcHEP

5. DESIGN AND IMPLIMENTATION

Pad++ is implemented in C++ under various versions of the UNIX operating system using the standard X graphics library system. It currently runs on SGIs, Suns, IBM RS-6000s, PCs running Linux, and should be trivially portable to other UNIX systems. Pad++ is implemented as a widget in Tcl/Tk and thus allows applications to be written in the interpreted Tcl language. All Pad++ features are accessible through Tcl making it unnecessary to write any new C++ code. The efficiency methods we use in Pad++ include:

• **Spatial Indexing:** Objects are stored internally in a hierarchy based on bounding boxes which allow fast indexing to visible objects.

• **Clustering:** Pad++ automatically restructures the hierarchy of objects to maintain a balanced tree which is necessary for the fastest indexing.

• **Refinement:** Render fast while navigating by using lower resolution, and not drawing very small items. When the system is idle for a short time, the scene is successively refined, until it is drawn at maximum resolution.

• **Level-Of-Detail:** Render items differently depending on how large they appear on the screen. If they are small, render them with lower resolution.

• **Region Management:** Only update the portion of the screen that has been changed. Linked with refinement, this allows different areas of the screen to refine independently.

. Clipping: Only render the portions of large objects that are actually visible. This applies to images and text.

• **Adjustable Frame Rate:** Animations and zooming maintain constant perceptual flow, independent of processor speed, scene complexity, and window size. This is accomplished by rendering more or fewer frames, as time allows.

• **Interruption**: Slow tasks, such as animation and refinement, are interruptible by certain input events (such as key-presses and mouse-clicks). Animations are immediately brought to their end state and refinement is interrupted, immediately returning control to the user.

• **Ephemeral Objects:** Certain objects that represent large disk-based datasets (such as the directory browser) can be tagged ephemeral. They will automatically get removed when they have not been rendered for some time, and then will get reloaded when they become visible again.

• **Optimized Image Rendering:** The code to render zoomed images has been very carefully optimized and allows real-time zooming of high-resolution images.

Scripting Language Interface

An important consideration in the design and implementation of Pad++ is how to create a very fast and efficient graphics system, and yet still make it extensible. I wanted to make sure that author and others would be able to easily experiment with new interface mechanisms. Originally, Pad++ was implemented entirely in C++, making it very difficult for anyone but the authors to add new objects and interactions. Even for the authors, going through the compile and link cycle was very slow and tedious, making it difficult to do much experimentation. I decided to create an interpreted scripting language interface to Pad++ to get around this problem. This approach is becoming quite common, and works well as long as the scripting language is at the right level. On one side, you want as much as possible to

be in the scripting language so that the system is as easy to modify as possible. On the other side, it is critical that all speed-critical code be written as efficiently as possible.

Physics-Based Strategies for Interface Design

I propose information physics view of interface objects that we think provides an effective complement to traditional metaphor-based approaches. While an informational physics strategy for interface design certainly involves metaphor, author says there is much that is distinctive about a physics-based approach. While there are ease-of-use benefits from such mappings, they also orient designers towards mimicking mechanisms of earlier media rather than towards exploring potentially more effective computer-based mechanisms. Semantic zooming is but one example mechanism that I think more naturally arises from adopting an informational physics strategy.

Simulations and constraint-based interfaces that led to the development of direct manipulation style interfaces are other examples of this general approach. They too derive from Sutherland and continue to inspire developments. More importantly, they are based on implementation of mechanisms at a different level than is traditional. It is important to look at the costs as well as the benefits of traditional metaphor based strategies. They can lead away from exploration of new mechanisms and limit views of possible interfaces in at least four ways.

First, metaphors necessarily pre-exist their use. Pre-Copernicans could never have used the metaphor of the solar system for describing the atom. In designing interfaces, one is limited to the metaphorical resources at hand. In addition, the metaphorical reference must be familiar in order to work. An unfamiliar interface metaphor is functionally no metaphor at all. One can never design metaphors the way one can design self-consistent physical descriptions of appearance and behavior.

Second, metaphors are temporary bridging concepts. When they become ubiquitous, they die. In the same way that linguistic metaphors lose their metaphorical impact successful metaphors also wind up as dead metaphors (e.g. file, menu, and window, desktop). The familiarity provided by the metaphor during earlier stages of use gives way to a familiarity with the interface due to actual experience. Thus, after a while, it is the actual details of appearance and behavior (i.e. the physics) rather than any overarching metaphor that form much of the substantive knowledge of an experienced user. Hierarchies.

Third, since the sheer amount and complexity of information with which we need to interact continues to grow, we require interface design strategies that *scale*. A traditional metaphor-based strategy does not scale. A physics approach, on the other hand, scales to organize greater and greater complexity by uniform application of sets of simple laws.

Fourth, it is clear that metaphors can be harmful as well as helpful since they may well lead users to import knowledge not supported by the interface. Our point is not that metaphors are not useful but that as the primary design strategy they may well restrict the range of interfaces designers consider and impose less effective tradeoffs than designers might come to if they were led to consider a larger space of possible interfaces. One might want to focus on easily discoverable physics. As is the case with metaphors, all physics are not created equally discoverable or equally fitted to the requirements of human cognition

The Physics Interface is the part of the interface that creates all objects in a world and the worlds their selves. The user starts using the interface by choosing which engine to use. For example by simply writing: **PhysicsInterface*myInterface=new NewtonInterface()** the user tells the interface to use the Newton Dynamics Engine and its functionality. The interface creates worlds (see below) and saves a pointer to them in a reference list. The interface class is mostly used internally by the worlds to create all sorts of objects, materials and joints using the create –method

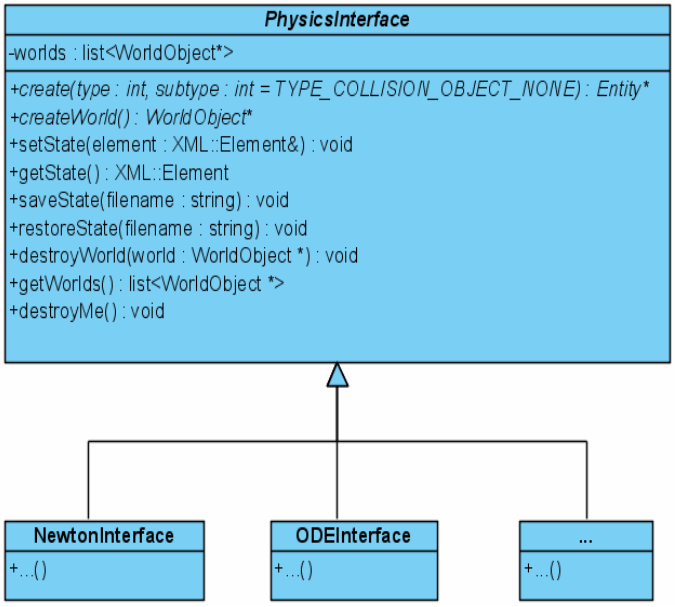

Figure 8. Physics Interface

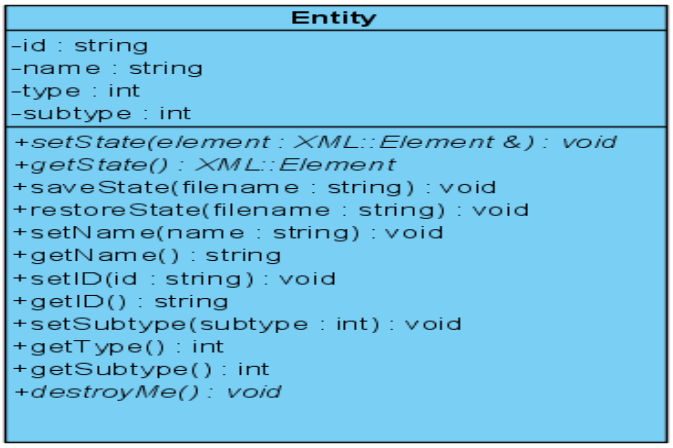

Figure 9. Entity

Created by the interface, the world is responsible for creating and destroying objects, materials and joints and storing pointers to all of them. To create a world, write: **WorldObject* world = myInterface->createWorld();**

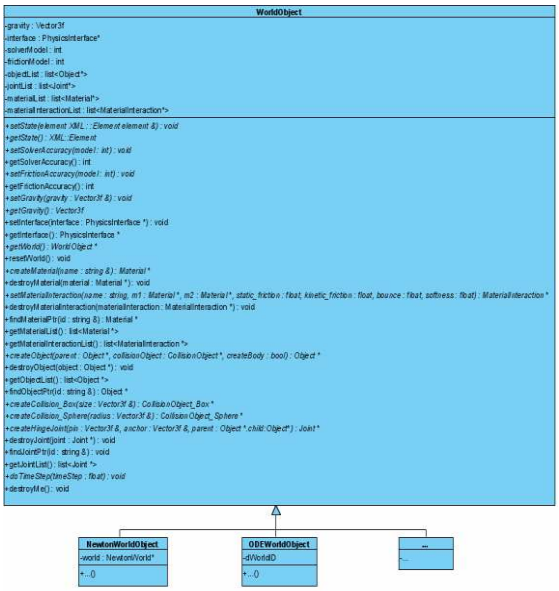

Figure 10. World Objects

Figure 11. Material interaction

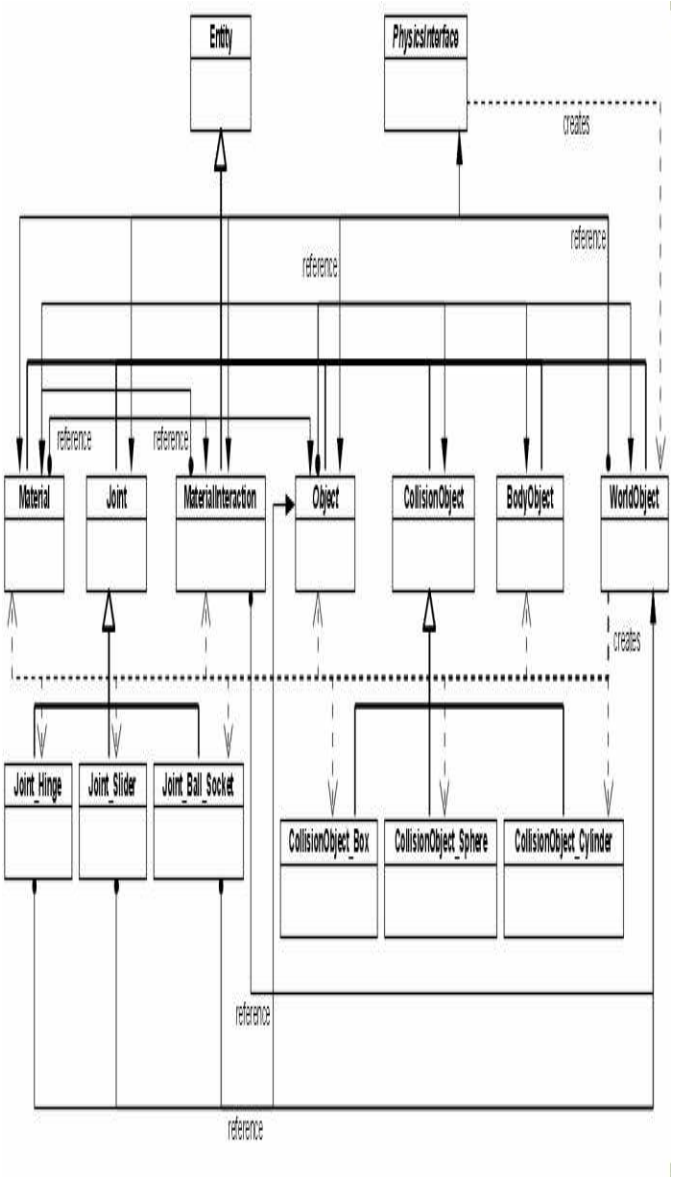

Figure 12. Interaction Diagram of Interface

6. NUMERICAL SESSION & WORK

Breit-Wigner propagator [18]

The propagator denominator of a particle at tree level is given by $1/p^2-m^2$ where m is the particle mass and has a pole at $p^2=m^2$ If this pole is inside the phase space volume being integrated over, it causes the integral to diverge. At higher order, the propagator denominator is modified to become $1/p^2-m^2-i\prod(p^2)$ m where $\prod(m^2)$ is the width of the particle (the inverse of the particle's mean lifetime). This removes the pole and renders the integral convergent. Since the $\prod(p^2)$ terms only dominates this propagator denominator near the pole (and is a small correction far from the pole), this propagator is well approximated by replacing $\prod(p^2)$ with $\prod = \prod(m^2)$ which gives the Breit-Wigner propagator denominator $1/p^2-m^2-i\prod m$. This is the propagator denominator used in CalcHEP.

There are three different regimes to consider. In the first regime, the particle is exactly on shell. In this regime the calculation is exactly gauge invariant and there is no problem. In the second regime, the particle is off shell, but not very far from on shell. In this regime, the process is still dominated by the resonant diagrams and the effect of gauge invariance breaking is still small. In the third regime, the width is not needed to regularize the integral. So, gauge invariance can be satisfied by not including the width. The Breit-Wigner menu allows adjusting the properties of the propagator denominators used in the phase space integrals.

The second menu item is T-channel widths and allows turning the width on in the t-channel propagators. The default is to not include these since they are not required to regularize the integration and these propagators never go on shell. Note that, by default, the symbolic session does not include the widths in the t-channel propagators. In order to turn this on, the user must also turn it on during the symbolic session. The last two menu items (GI in t-channel and GI in s-channel) control whether CalCHEP applies another method of restoring gauge invariance. The diagrams which do not contain the resonant propagator are multiplied by the factors $(p^2-m^2)/(p^2-m^2)2+(\prod m)^2$ expression This modification corresponds to the symbolic summation of all diagram contributions at a common denominator expression with subsequent substitution of the width term into the factored denominator. The trick allows keeping all gauge-motivated cancellations As a defect of the trick it should be mentioned that the factor $(p^2-m^2)/(p^2-m^2)2+(\prod m)^2$ kills contributions of non-pole diagrams in the $p^2 = m^2$ point.

Event Generation [19]

CalcHEP can generate unweighted events. These events are useful in simulations of particle physics collisions and can be passed to other programs for further analysis. Event generation is done in the Monte-Carlo simulation [20] menu. It consists of two steps. In the first step, the generator is prepared and in the second, the events are generated. The efficiency of the event generator depends on the number of phase space cubes and the estimation of the maximum differential cross section (or partial width) in each cube. Since the differential cross section (or partial width) can vary greatly from phase space point to point (especially around a resonance), a larger number of phase space cubes allows for a more efficient generator. The menu item Event cubes allows modifying this parameter.

On the other hand, for each phase space cube, it is very important to have a good estimate of the maximum for that cube. This process can be continued until a satisfactory efficiency is achieved. The user must balance obtaining a high efficiency against the time it takes to estimate the maxima for a large number of phase space cubes. When the generator is prepared and the efficiency is acceptable, the user can enter the Generate Events menu under Monte-Carlo Simulation.

The first item on this menu, Number of events, allows specifying the number of events to generate. Event generation is started by the menu item Launch generator. During event generation, if an event is ever produced with a differential cross section (or partial width) which is greater than the estimated maximum value in that cube then two things happen. The first is that the event is split into multiple unweighted events.

The second is that the maximum for that phase space cube is increased in order to prevent this from occurring in the future. When CalCHEP finishes generating the events, it displays an informational message which states the number of events generated, the structure functions are not included, the center-of-mass rapidity is set to zero, the regularizations are ignored and all the cuts are ignored. In fact, the only cut allowed in the Simpson integration is on the cosine of the angle between the third particle (the first outgoing particle) and the first incoming particle. We call this cos13. The user can modify the minimum and maximum values of this kinematical variable via the Cos13 (min) and the Cos13 (max) menu items. This cut is often necessary to remove T-channel singularities from mass less propagators (such as a T-channel photon in e+e+→ e+e+) [20]. The default cut is-0.999 < cos13 < 0.999. Upon successful integration, the cross section is displayed on the screen. The precision of the calculation can be set via the Set precision menu item. The default value is 10^{-4}. The angular dependence of the differential cross section can be viewed via the Angular dependence menu item.

Plots of the dependence of the cross section on the model parameters or the center of mass energy can be viewed by the Parameter dependence menu item. This menu also provides sigma*v plots for $v \times \sigma(v)$ at $v \to 0$. Here v is relative velocity of particles. This characteristic is useful for astroparticle applications, in particular for the estimations of the elastic scattering of Dark Matter candidates [21] .

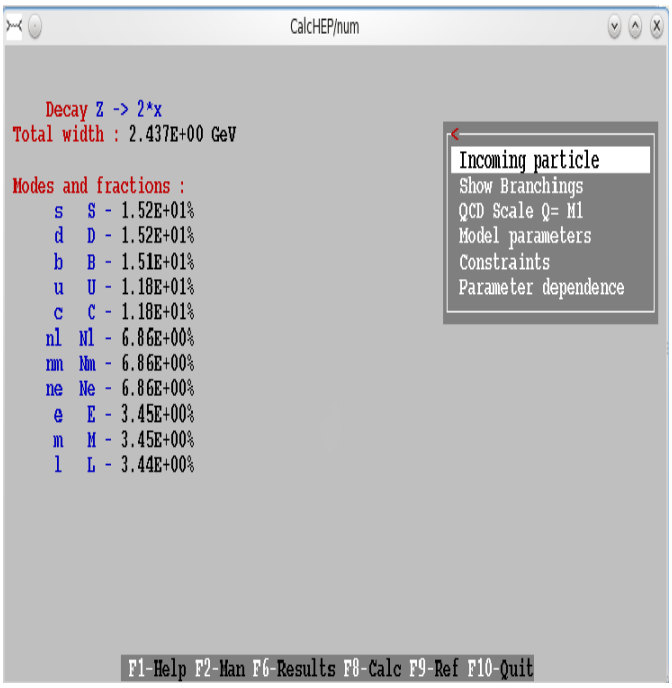

Figure 13. Two Particle Decay at SM Z Boson

When connecting decays, event_mixer uses a Breit-Wigner virtual mass distribution, where we assume that the matrix elements of the sub processes do not depend

strongly on the off-shell momentum. My procedure does not break momentum conservation. According to the LHE file format accord, the header (marked by <header> and </header>) section can be used for auxiliary information. event_mixer places the following in the header: a <hepml> section, a <slha> section with information about quantum numbers, masses widths and decays of non-SM particles, and a <calchep-batch> section for the 'run details.txt' file. Information about the process such as a list of the sub processes, kinematical cuts, model name, and number of generated events, cross sections and model parameters is stored in the <hepml> section. For instance,

```
    <files>
  <file>
      <eventsNumber> 1000 </eventsNumber>
    <crossSection unit="pb"> 0.254087
</crossSection>
  </file>
</files>
```

As the number of final state particles increases, it can be very helpful to specify the kinematics which helps CalCHEP in the numerical integration stage. This is done in exactly the same notation as in CalCHEP. The numbering corresponds to the order the particles are entered in the process in the batch file.

This example generates 1000 events (for each Mh) of the process p, p->W, b, B for the model Standard Model (CKM=1).

```
   Model Info
###################################
Model: Standard Model (CKM=1); Model changed: False; Gauge: Feynman
###################################
 Process Info
Process: p,p->W,b,B; Decay: W->le,n
Composite: p=u,U,d,D,s,S,c,C,b,B,G
Composite: W=W+, W-; Composite: le=e, E, m, M
Composite:n=ne,Ne,nm,Nm;Composite:
jet=u,U,d,D,s,S,c,C,b,B,G
###################################
# Momentum Info
###################################
p1: 4000, p2: 4000
```

```
# Parameter Info
###################################
Parameter: Mtp=172.5
###################################
# Run Info
###################################
Run parameter: Mh
Run begin: 120
Run step size: 5
Run n steps: 3
#####################################
# QCD Running Info
#####################################
Alpha Q: M45
#####################################
# Cut Info
#####################################
Cut parameter: M (b, B)
Cut invert: False
Cut min: 100
Cut max:
Cut parameter: J (jet, jet)
Cut invert: False
Cut min: 0.5
Cut max:
Cut parameter: T (jet); Cut invert: False
Cut min: 20; Cut max:
Cut parameter: N (jet); Cut invert: False
Cut min: -2.5; Cut max: 2.5
#####################################
# Kinematics Info
#####################################
Kinematics: 12 -> 3, 45; Kinematics: 45 -> 4, 5
#####################################
# Regularization Info
#####################################
Regularization momentum: 45; Regularization mass: Mh
Regularization width: WH; Regularization power: 2
#########################################
# Distribution Info
#################################
Dist parameter: M (W+, b), Dist min: 100
Dist max: 200; Dist n bins: 100
Dist title: p, p->W, b, B; Dist x-title: M (W+, b) (GeV)
###############################
```

7. CONCLUSION & FUTURE WORK

This report gives an explanation to what a physics engine consist of and how it works. I have not concentrated so much on algorithms used in a physics engine or how the algorithms work. For the interested reader, algorithms and implementation code, to name a few. Instead, author has given an explanation to what the physics engine do that may be easy to understand for most readers. There are many ways to implement a physics engine and author present an overlook of how a typical physics engine work. An important part in the physics engine is the constraints, used for collision response and joints. This part of the paper was for getting a good basic knowledge of physics engines and to give a better understanding for the rest of the paper. Since a physics engine can simulate more than just rigid bodies, I described in short how deformable objects can be modeled and how particle systems work; this was to describe what a physics engine is capable of. I chose a few physics engines and made a short overview of them, showing what their functionalities consisted of and chose two of them for a closer look. All functionality in this interface depends on the engine used.

If this work is taken in R&D with crustier, this can give a new concept on technology like explosive detection of planted or implanted via satellite. It can also give new path in nuclear development. This interface can also give a new path in internet service to increase it bandwidth of 2 GB/s data flow. And many more scope which can give

a new question mark on technology. Very important through this interface it is clear that there is no black hole in universe. My next publication will be in Black hole, describing its presence and advantage.

8. REFERENCES

[1]. A. Pukhov, CalcHEP 2.3: MSSM, structure functions, event generation,batchs, and generation of matrix elements for other packages (2004).arXiv:hep-ph/0412191.
[2]. F. Yuasa, et al., Automatic computation of cross sections in HEP: Status of GRACE system, Prog. Theor. Phys. Suppl. 138 (2000) 18–23.arXiv: hep-ph/0007053.
[3]-[4] Quantum computer ,a theoretical approach book, written by Author of this paper limted edition only for R&D.
[5]. ProQuest Dissertations And Theses; Thesis (Ph.D.)--Harvard University, 2009.; Publication Number: AAI3365296; ISBN: 9781109254976; Source: Dissertation Abstracts International, Volume: 70-07, Section: B, page: 4243.; 245 p
[6]. Thomsan book of black body radation and quantum theory in collected paper of enstien vol 2
[7]-[8] This article is taken from author own book and practical study.
[9].wikipedia ,www.google.com;
[10]. E. Boos, et al., CompHEP 4.4: Automatic computations from Lagrangians to events, Nucl. Instrum. Meth. A534 (2004) 250–259.arXiv:hep-ph/0403113, doi:10.1016/j.nima.2004.07.096.
[11].http://www.fizika.unios.hr/~ilukacevic/dokumenti/materijali_za_studente/qm2/Lecture_2_Perturbation_theory.pdf
[12]. A. Pukhov, CalcHEP 2.3: MSSM, structure functions, event generation, batchs, and generation of matrix elements for other packages (2004). arXiv:hep-ph/0412191.
[13]. A. Pukhov, et al., CompHEP: A package for evaluation of Feynman diagrams and integration over multi-particle phase space. User's manual for version 33 (1999). arXiv:hep-ph/9908288.
[14]. G. Belanger, et al., Automatic calculations in high energy physics and Grace at one-loop, Phys. Rept. 430 (2006)117–209.arXiv:hep-ph/0308080 doi:10.1016/j.physrep.2006.02.001.
[15]. T. Hahn, Generating Feynman diagrams and amplitudes with FeynArts 3, Comput. Phys. Commun. 140 (2001) 418–431. arXiv:hep-ph/0012260, doi:10.1016/S0010-4655(01)00290-9.
[16]. F. Maltoni, T. Stelzer, MadEvent: Automatic event generation with MadGraph, JHEP 02 (2003) 027. arXiv:hep-ph/0208156.
[17]. G. Belanger, F. Boudjema, A. Pukhov, A. Semenov, micromegas: A program for calculating the relic density in the mssm, Comput. Phys. Commun. 149 (2002) 103–120. arXiv:hep-ph/0112278.
[18]. T. Gleisberg, et al., SHERPA 1.alpha, a proof-of-concept version, JHEP 02 (2004) 056. arXiv:hep-ph/0311263, doi:10.1088/1126-6708/2004/02/056.
[19]. T. Gleisberg, et al., Event generation with SHERPA 1.1, JHEP 02 (2009)007. arXiv:0811.4622, doi:10.1088/1126-6708/2009/02/007.
[20] H. Tanaka, T. Kaneko, Y. Shimizu, Numerical calculation of Feynman amplitudes for electroweak theories and an application to e+e- ! W+ W- gamma, Comput. Phys. Commun. 64 (1991) 149–166.doi:10.1016/0010-4655(91)90058-S
[21] G. Belanger, et al., Indirect search for dark matter with micrOMEGAs2.4, Comput. Phys. Commun. 182 (2011)842–856.arXiv:1004.1092, doi:10.1016/j.cpc.2010.11.033.
J. A. M. Vermaseren, Axodraw, Comput. Phys. Commun. 83 (1994) 45–58. doi:10.1016/0010-4655(94)90034-5.

NITISH KUMAR- Has received highest degree in physics and currently pursuing degree in computer science, an extension course with respect to his R&D work. Author has also received membership in various journals like ACM, CASTA IRAJ, UACEE-USA AND THOMSON RECIUTER. Author is presently a member of ABVP.

Author has published more than 8 international journals in computer science, and also submitted project in department of science in technology (New Delhi). His keen interest of research is on QUANTUM COMPUTATION AND ITS APPLICATION RELEATED TO PHYSICS & MACHINE LEARNING PROCESS.