Procedural Animation of 3D Humanoid Characters Using Trigonometric Expressions

Zeeshan Bhatti, Imdad Ali Ismaili, Shehnila Zardari, Hafiz Abid Mahmood Malik, Mostafa Karbasi

***Abstract* – This paper discusses a technique of generating procedural animation of bipedal characters through expressions in Maya using trigonometric functions. The character skeleton of bipeds is procedurally generated using a template based widgets mechanism. The skeletal joints created and rigging is done through a procedural programming technique based on the underlying widget structure. The expressions using trigonometric functions are then applied on each body part to produce procedural motion. The entire system is procedurally generated and the user is presented with a complete rigged character with simulation. The various parameters of the animation are then controlled by and easy to use graphical user interface, which allows the animator to further manipulate and customize the animation. All the changes in the parameters are reflected in the simulated skeleton in real-time. Using these trigonometric equations, various types of biped motions are generated and controlled at runtime dynamically, producing a wide range of animations which can be easily customized and further extended according to the requirements of the scene.**

Index Terms – Animation, Biped, Procedural motion, Simulation.

I. INTRODUCTION

The vastly growing movie and game industry with ever increasing demand of realistic animation of 3D characters has increased the challenges posed to a 3D artist. There is an ongoing debate among the leading animators and production studio over quality verses quantity of the animation. As the amount of projects increase every year and human resource of qualified animators is scarcely available, the production studios harshly demand a large quantity of animation produced in a short time. This is one of the major reason that most of the large production studios are forced to use motion capture for producing realistic animation quickly and accurately.

The key-frame being a traditional approach takes a huge time to render a realistic looking animation. There is a big time loss, but with motion capture the time is reduced, consequently increasing other complications like the cost of motion capture sensors, specially trained personals, hardware and software requirements etc. Finally the procedural animation comes into the picture. With the advent of technology and algorithms it seems a lot easier to write an algorithm that would simply move a character according to its biological gaits with locomotion parameters and producing a realistic looking animation with a single click of a button. Obviously this approach also has various unfeasible constraints that limit its usability and practical implementation. First, the algorithms to produce a realistic motion are very complex especially when physics based dynamic motions are to be achieved. Algorithms are then usually tested and implemented in a special environment using particular tools like SIMBICON [1], Open Dynamic Engine (ODE) [2] are but few. Application of these tools is good for research purpose, however, in practice it is difficult to implement these algorithms in professional software like MAYA. Hence, most of this work is limited to research only, rather than actual implementation. The major setback of such procedural animation methodologies is that these are hardly customizable. The animator is unable to change and modify a large part of the motion that is obtained procedurally and is stuck with what the programmer has programed the 3D character to do. In this project we have addressed this issue of practical usability, simplicity of motion equations and customization of procedurally generated animation.

The area of biped simulation is extensively researched and discussed in the computer graphics industry due to its importance in animation, robotics and biomechanics. Various algorithms and techniques have been addressed for generating realistic looking simulation or animation of humanoid characters locomotion's. The problem of developing physics based kinematics or dynamic motions models for generating complex human motion is still considered a challenging problem [3].

Laszlo et al. has developed an interactive user-in-the-loop technique for controlling physics based animation by interactively controlling the planer dynamic simulation using a mouse and keyboard [3]. Whereas, I. D. Horswill uses kinematics based model through procedural animation to produce realistic dynamic motion with each body part being controlled by external force instead of using joint torques [4]. However, [5] uses prerecorded motion capture data to reproduce the animation of human locomotion with combination of physics based system thus creating a hybrid approach. While Yin et al. in their remarkable work on SIMBICON, uses simple finite state machine and poses control graph with each joint angle controlled using Proportional Derivative(PD) to produce a large variety of biped locomotion gaits and styles in real-time [1]. The remarkable robustness of SIMBICON project was further enhanced with controller adaptation by Yin et al. [6]. Wang, Suwen[7] also optimized SIMBICON to achieve realistic results by using biomechanically driven functions with more human like gaits . Coros et al. [8] developed a control strategy to simulate walking motions of biped using proportional-derivative control with foot placement technique based on an inverted pendulum model. The

Zeeshan Bhatti and Imdad Ali Ismaili are with Institute of Information and Communication Technology, University of Sindh, Jamshoro. Shehnila Zardari is with Department of Computer Science & Information Technology, NED University of Engineering & Technology, Karachi. Hafiz Abid Mahmood Malik and Mostafa Karbasi are with Department of Computer Science, International Islamic University Malaysia.
Email: zeeshan.bhaati@usind.edu.pk, iai_a@yahoo.com, shehnilaz@neduet.edu.pk, abid.malik@live.iium.edu.my, mostafa.karbasi@live.iium.edu.my.
Manuscript received Dec 13,2016; revised on Dec 23, 2016; accepted on Dec 27, 2016.

 ISSN – 1999-4974

physics of gravity and velocities errors with virtual forces were adjusted using Jacobian transpose control. The use of an inverted pendulum model (IPM) was used in [8] for biped animation where authors used generalization according to the character size. However, the two main problems with using physics passed simulation systems are also addressed in [9].

II. SYSTEM DESIGN AND ARCHITECTURE

The mathematical expression component is based on trigonometric equations that consists of sine and cosine function with control parameters. This mathematical model and formula has been adopted from the work of [10][11][12]. The expressions are applied on each body part individually to generate periodic motion for each gait. The user input is provided through the GUI that derives and controls the gait behaviour during runtime as discussed in [13]. The resulting motion curve is summed over and applied on the dynamic dual layer based quadruped rig for final output. The overview of this animation system based on procedural modelling is shown in Figure 1.

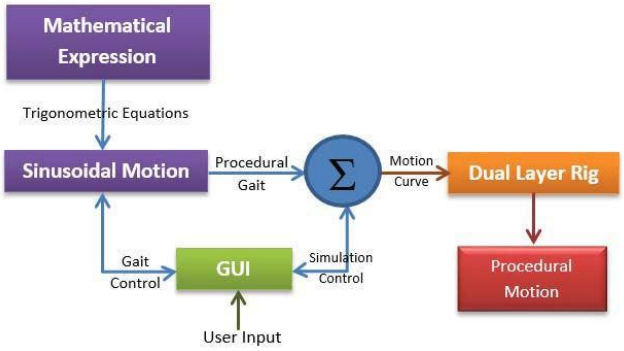

Fig. 1 Overview of Procedural Animation System [12]

A. *Sine Function Justification*

In order to develop a basic and simple motion equation, we started with trigonometric sine function. As the primary aim was to develop a procedural animation methodology which is easily implemented, fast and simple to use, thus the sine function was a perfect choice for generating the motion curves with oscillating phase. The sine function characteristics allow the motion frequency to be controlled with varying amplitudes and phases of the motion curve as discussed in [12][11]. Therefore, as the natural skeletal joint trajectories are simple sinusoidal patterns, thus the oscillating motion of leg joints follow this sine wave approximately. This nominal stepping sine wave is then controlled using the coupled oscillator model- adopted form [14][15] with Central Pattern Generator system extended from the work of [16], implemented on biped locomotion using non-linear oscillators.

III. BIPED EXPRESSIONS

For generating the procedural motion of a biped character, the skeletal rig has been segregated into five main sections [17][18]. The five sections consist of Chest/neck, Arms, Legs and Pelvis/Spine sections. The expressions of each body section is calculated and implemented separately. The forward motion of the biped characters is calculated from the basic trigonometric sine and cosine functions. Initially three main parameters were defined and used to calculate the torque in forward direction. The motion velocity (*vM*) controls how fast or slow the character moves during various gaits. By changing this parameter alone the motion gaits of character is altered at runtime. The time 'T' parameter is time in frames per second and is obtained from the Maya system. Finally, the phase of angular motion 'ϕ' is used to alternate the phase of the motion frequency.

A. *Pelvic and Spine Expressions*

The pelvic region along with the entire spine is the key part of the body that determines and gives a solid stability to the entire body during motion. Various parameters have been used here to control and customize the motion of pelvic or hip region.

Equation (1) produces the basic movement of the pelvic joint in y-direction. This pelvic region behaves as the main center of mass of biped and thus is the root of our character.

$$f(\text{pelvis}) = \sin(T * vM * \phi) * \omega B \quad (1)$$

Where, T is the Time in frames > 1, *vM* velocity of the forward Motion which is always > 0 and ϕ is the phase of angular motion. The angular frequency of body oscillation is controlled by ωB parameter. These four variables are used in conjunction with trigonometric sine functions to generate the initial cyclic equations used for biped motion.

Similarly, the chest oscillation in forward and backward direction is controlled using the 'ωC' variable in (2). The equation (2) generates the motion of upper body involving spine, chest and shoulders.

$$f(\text{spine}) = \sin(T * vM * k) * \omega C \quad (2)$$

Where, '*k*' is a constant whose value is between 0 and 5, used here for optimization purpose and is adjusted according to the size of the character.

The motion of hip oscillation is controlled by Hosc, whereas the height and position of the hips section is manipulated using the Hhgt and Hpos attributes.

The attributes to control shoulder motion are also similar yet separately defined as Sosc is used for Shoulder Oscillation. The Shgt and Spos are used to control the Shoulder height and position with respect to chest joint. Now the transformation of each spine deformer (the cluster) is driven using the mathematical expressions, combining the custom attributes with motion equations discussed in previous section. The cluster handle located near the shoulder joint (StY and StZ) is translated in y-axis and z-axis along with shoulder height and body oscillating parameter. The J^{error} constant is used for error reduction with value of 0.85.

ISSN – 1999-4974

$$StY = Shgt + Tz * \omega B - J^{error} \quad (3)$$

$$StZ = Tz * Sosc – Spos – f(spine)- Hpos \quad (4)$$

The Tz parameter is the Translate in z-axis obtained from equation (9) and Sosc is Shoulder Oscillation attribute. In order to modify the above equation for base cluster control, we use (4) with body oscillation and hip position to translate the cluster in z-axis. For translation in y-axis, (4) is multiplied with hip oscillating attribute and divided with a constant value of 4 to reduce the overshoot error. Further equation (1) is used along with hip height (Hhgt) and J^{error} with value 0.523 used for optimization.

$$CtZ = Tz * \omega B – Hpos \quad (5)$$

$$CtY = Tz * Hosc / 4 + f(pelvis) + Hhgt - J^{error} \quad (6)$$

The main spine IK controller which is located near the shoulder cluster control is translated with the shoulder cluster with minor modification of the value to reduce the error and optimize the animation control. Hence, the J^{error} attribute is introduced with value 0.8 for translation in y-axis and 0.02 for z-axis respectively.

$$StY\ (ik) = StY + J^{error}$$
$$StZ\ (ik) = StZ - J^{error} \quad (7)$$

B. Legs and Feet Expressions

The motion of the legs is the most significant part of biped locomotion. The timing curve and torque of each leg joints determines the gait type of the character during motion.

The distance of foot from the ground is measured using ‘Δf’ in (8) that determines distance of the foot from the ground while in motion. Using this variable we control how close or high the foot must be from the ground during the motion (for example when cycling the foot must be very high as compared to walking). The equation (8) and (9) calculate the basic translation values in ‘y’ (Ty) and ‘z’ (Tz) axis respectively.

$$ty = \sin (T * vM) + \Delta f \quad (8)$$

$$tz = \cos (T * vM) \quad (9)$$

For maximum control over the animation we have used separate parameters and expression to control each leg individually. The parameters used for controlling the foot height from the ground during motion are $Fhgt^R$ & $Fhgt^L$ for right and left legs respectively. The $Fpos^R$ & $Fpos^L$ control the position of foot with respect to the root of body. The attribute to controls the length of the stride are $Slgt^R$ & $Slgt^L$. All these attributes are user defined and controlled by the user through easy to use interface. These attributes affect the amplitude of the motion and alters the gaits at runtime.

For optimizing the error correction in floor contact phase of foot, (1) is modified and fine-tuned to (10). In (10) the value of α is set to 70 and the J^{error} for our joint error optimization is set to 20. This value may vary for different characters depending on the size of leg (from up_leg joint to ankle joint) and also the size of foot (from ankle joint to toe_end joint). So some manual tweaking is required for different characters. The Fτ in (5) then is used to generate accurate transformation of foot IK handle and for left foot the equation is reversed by simply negating its value.

$$F\tau = \sin (T * vM - k) / J^{error} \quad (10)$$

The expression driven implementation for involuntary leg motion is given below. The expressions are now simple mathematical calculations using the equations and parameters discussed previously. The basic translation of each IK based foot controller, in y-axis is achieved using (8) with foot height parameter of each leg. For translation in z-axis, (9) is multiplied with stride length and foot position. This gives us the forward motion of each leg depending on the length of stride and the position of foot during each gate.

$$RightFoot\_CTRL.translateY = Ty * Fhgt^R \quad (11)$$
$$RightFoot\_CTRL.translateZ = -Tz * Slgt^R + Fpos^R$$

$$LeftFoot\_CTRL.translateY = (-Ty) * Fhgt^L; \quad (12)$$
$$LeftFoot\_CTRL.translateZ = -Tz * Slgt^L + Fpos^L$$

The toe of each foot is individually controlled using (10) translation, in y-axis of the Toe_IK controller. This is again done on toe_ik controller of each leg separately.

$$RightToe\_IK.tY = F\tau$$
$$LeftToe\_IK.tY = - (F\tau) \quad (13)$$

Finally the expression for right up-leg joint motion is implemented using (9) with hip oscillation and Hip position attributes to add flexibility at pelvic region during the motion. The z-axis of the up-leg joints is driven using (1) with (8) as shown below.

$$RightUpLeg.tZ = tz * Hosc - Hpos \quad (14)$$
$$RightUpLeg.tY = Hhgt + f(pelvis) – ty * Hosc$$

$$LeftUpLeg.tZ = -Tz * Hosc – Hpos \quad (15)$$
$$LeftUpLeg.tY = Hhgt + f(pelvis) - ty * Hosc$$

C. Shoulders and Arms Expressions

Arms are the most expressive part of human body. Their motion is usually independent of any gait behavior and usually unique to each character personality. The cyclic swing motion of arm has few additional attributes to cater such type of personality traits. Initially we control the bounciness of the arm from shoulder to the wrist joint using the $Abnc^R$ and $Abnc^L$ attribute for each right and left arms respectively. As with the legs, each arm is also controlled and manipulated separately from each other to increase the naturalness and provide advance level of control to the arm motion. Therefore, the frequency of swing motion of each arm is controlled using the $\omega_{(ra)}$ & $\omega_{(la)}$ parameters. The position of each Wrist motion in forward and back direction

ISSN – 1999-4974

with respect to the shoulder joint is controlled by $Wpos^R$ and $Wpos^L$. The standard length of arm in relaxed pose is measured by wrist being near the hips or pelvic region. The $Whgt^R$ and $Whgt^L$ determine the highness of the wrist with respect to the pelvic joint. By modifying this parameter the user can control the location of wrist to make it near the pelvis or near the chest or face or even high above the head. This change will be updated in real-time. Finally the $Wosc^R$ and $Wosc^L$ attribute is used to control the overall oscillation of the wrist. Hence, by using these attributes an advance level of control can be achieved with complete customization according to the requirements of the animator.

The basic swing motion of hand is generated using (16). Where the torque is calculated using arm bounce attribute and time, with sine function. Just like legs motion of each arm is also calculated individually. The value of constant α is set to 2 here.

$$A\tau^R = \sin (time * 2 * Abnc^R);$$
$$A\tau^L = \sin (time * 2 * Abnc^L); \quad (16)$$

The IK handle based controllers of each arm rig is translated in y-axis using (1) along with the arm height and oscillation attributes. The translation of IK controller in z-axis is achieved by using (9), arm motion frequency and position attributes.

$$RightArm\_IK.tY = Ahgt^R - A\tau^R * Aosc^R \quad (17)$$
$$RightArm\_IK.tZ = -Tz * \omega_{(ra)} + Apos^R$$

$$LeftArm\_IK.tY = Ahgt^L - A\tau^L * Aosc^L \quad (18)$$
$$LeftArm\_IK.tZ = Tz * \omega_{(la)} + Apos^L$$

The expression of shoulder involves multiplying (8) with (2) for y-axis and multiplying (9) with (2) for z-axis as shown below.

$$RightShoulder.tY = - Ty * f(spine) \quad (19)$$
$$RightShoulder.tZ = Tz * f(spine)$$

$$LeftShoulder.tY = Ty * f(spine) \quad (20)$$
$$LeftShoulder.tZ = - Tz * f(spine)$$

IV. RESULTS AND DISCUSSIONS

A simple to use graphical User Interface (GUI) was created having simple sliders for manipulating the parameters of the equation discussed previously. Modifying these attribute will affect the behavior of the biped motion at real-time as the motion is generated using expressions inside MAYA. Each body part is controlled and manipulated individually.

The Height of Shoulder motion is controlled using a High parameter as shown in Figure 2 (From left to right, high=0.6, high=0.2, high=0.807). When the height of shoulder is set to a low value then the character body is also pulled down and almost a crouching motion is created.

Similarly, a position parameter shown in Figure 3, controls the forward and backward position of the shoulder-chest. With default normal position is set to 0.

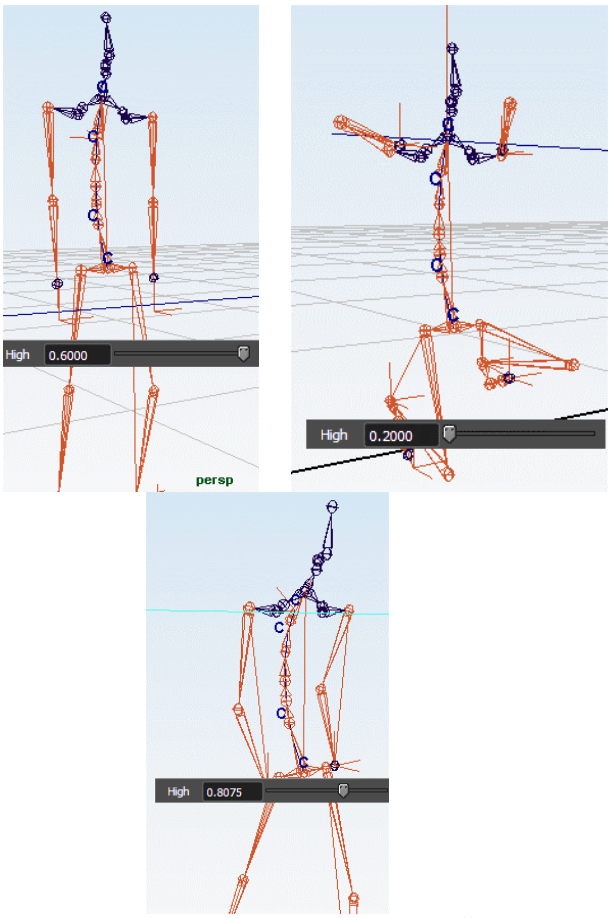

Fig. 2 Shoulder Height Parameter Effect

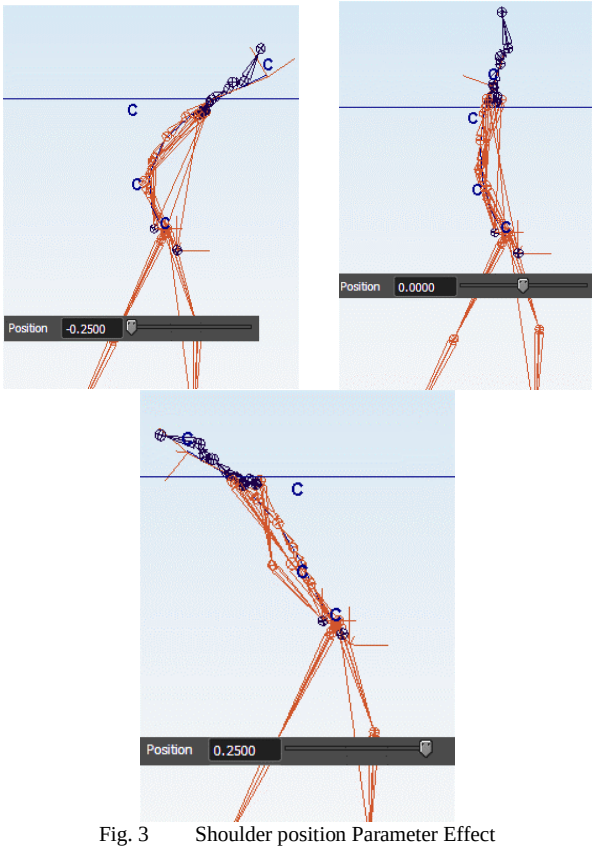

Fig. 3 Shoulder position Parameter Effect

ISSN – 1999-4974

The foot ground parameter controls the distance and height of each stride from the ground shown in Figure 4. This parameter is particularly useful when character is climbing stairs or cycling.

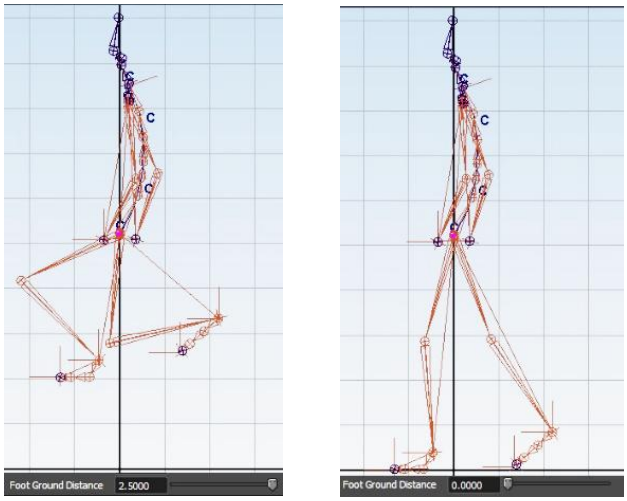

Fig. 4 Foot Ground Distance Parameter Result

Whereas, the height of the arm motion is controlled using the Highness parameter as shown in Figure 5.

Similarly, another parameter called Front controls the forward and backward position of the arm motion as shown in Figure 6.

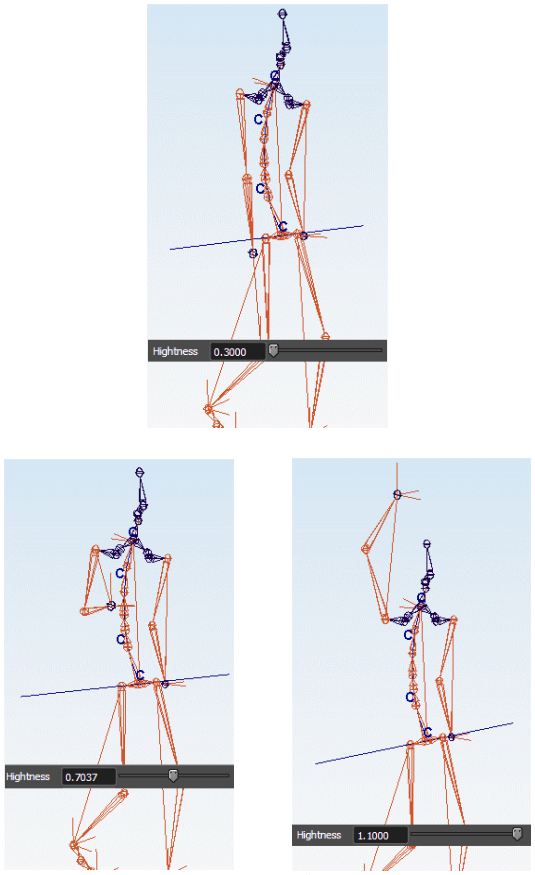

Fig. 5 Height of Arm

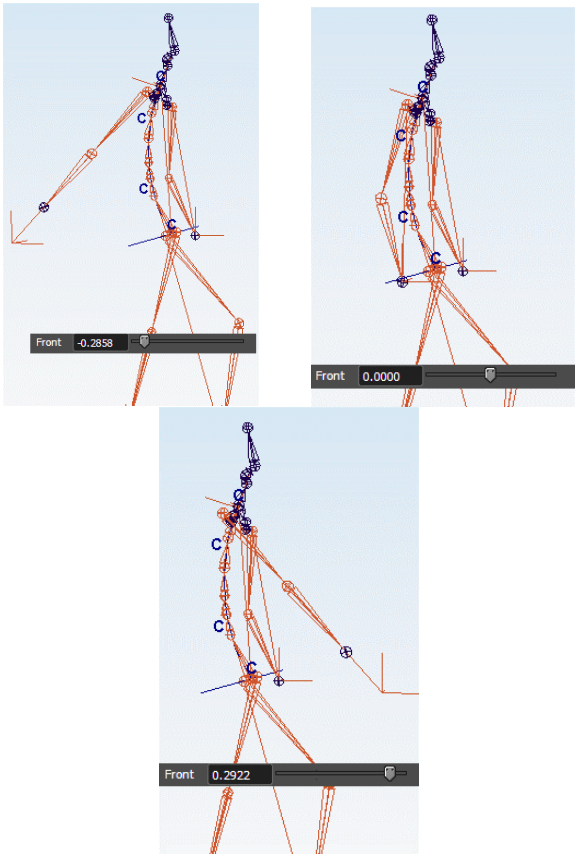

Fig. 6 Front Position of Arm

Whereas, the foot stride height is controlled using the Height from Ground parameter as shown in Figure 7.

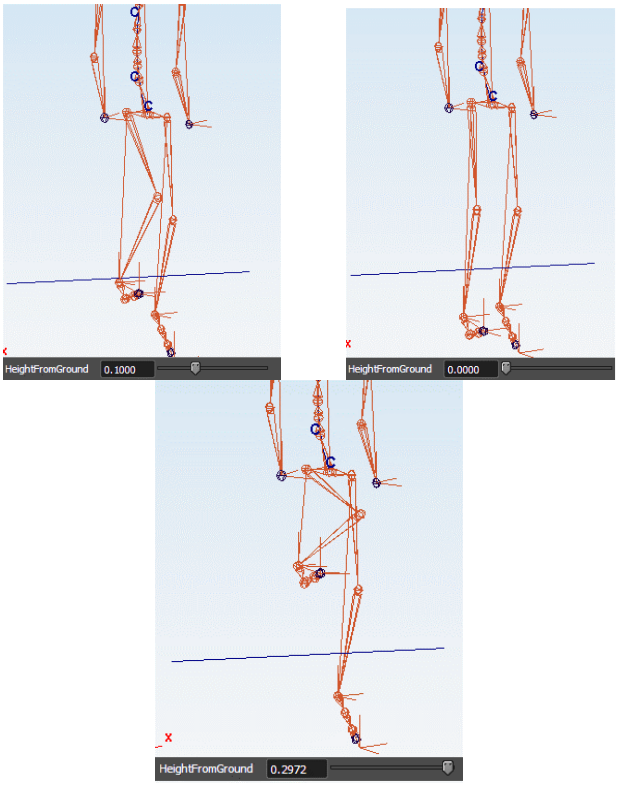

Fig. 7 Height of leg from ground

ISSN – 1999-4974

The final walk cycle simulation is shown in Figure 8, where a standard biped character is shown walking with natural gait.

Fig. 8 Walk Cycle

Alternately, a slightly abnormal motion is shown in Figure 9, where the character is walking with crouching gait.

Fig. 9 Walk with Feet High from the ground

V. CONCLUSION AND FUTUREWORK

In this paper a mathematical approach to generating procedural animation had been discussed using expression in MAYA. The mathematical equations were based on geometric trigonometric functions on procedurally generate rigged character skeleton. Each body part was individually controlled through mathematical expressions applied on the body controllers. The entire system is procedurally generated and the user is presented with a complete rigged character with simulation. The final results of biped walk motion is easily generated with few click and is highly believable. By altering few parameters, more variations in biped motion can be achieved.

There still remains substantial enhancements to be done in the system for the motion to be extremely realistic with multiple gait locomotion's. The equations need to be further optimized and tweaked to get best results. Moreover the equations can be enhanced and extended to generate more motions like run, jog, jumping and climbing etc. furthermore the gait transition has to be corrected and optimized for accurate and smoother transition from one gait type to another during runtime.

Appendix – A

List of Symbols

	Symbol	Description
1.	vM	Velocity of the forward Motion
2.	f	Frequency
3.	v	Velocity
4.	T	Time
5.	f(pelvis)	Frequency of Pelvic Joint
6.	ϕ	Phase
7.	ω	Angular Frequency / Oscillation
8.	ωB	Angular Frequency of Body
9.	d	*Frequency of Spine*
10.	k	Constant
11.	ωC	Angular Frequency of Body
Hip and Spine parameters		
12.	Hosc	Hip Oscillation
13.	Hhgt	Hip Height
14.	Hpos	Hip Position form Ground
15.	Sosc	Spine Oscillation
16.	Shgt	Spine Height
17.	Spos	Spine Position form Ground
18.	*StY*	Shoulder Translate in Y-axis
19.	*StZ*	Shoulder Translate in Z-axis
20.	J^{error}	Joint Error optimization
21.	CtY	Chest Translet in Y-axis
22.	CtZ	Chest Translet in Z-axis
23.	StY (ik)	Spine IK-handle Translate in Y-axis
24.	StZ (ik)	Shoulder IK-handle Translate in Z-axis
Leg Parameter		
25.	ty	Translate in y-axis
26.	Tz	Translate in z-axis
27.	Δf	Distance of Foot from Ground
28.	$Fhgt^R$	Right foot height from the ground during motion
29.	$Fhgt^L$	Left foot height from the ground during motion
30.	$Fpos^R$	position of Right foot with respect to the root of body
31.	$Fpos^L$	position of Left foot with respect to the root of body
32.	$Slgt^R$	Right leg Stride Length
33.	$Slgt^L$	Left leg Stride Length
34.	Fτ	Foot Translate
Arm Parameters		
35.	$Abnc^R$	bounciness of Right arm
36.	$Abnc^L$	bounciness of Left arm
37.	$\omega_{(ra)}$	frequency of Right arm swing Motion
38.	$\omega_{(la)}$	frequency of Left arm swing Motion
39.	$Wpos^R$	position of Right Wrist
40.	$Wpos^L$	position of Left Wrist
41.	$Whgt^R$	highness of Right Wrist
42.	$Whgt^L$	highness of Left Wrist
43.	$Wosc^R$	oscillation of Right wrist
44.	$Wosc^L$	oscillation of Left Wrist

ISSN – 1999-4974

REFERENCES

[1] K. Yin, K. Loken, and M. Van De Panne, "SIMBICON : Simple Biped Locomotion Control," ACM Trans. Graph., vol. 26, no. 3, p. 105, 2007.

[2] B. Kenwright, "Real-Time Physics-Based Fight Characters," no. September, 2012.

[3] J. Laszlo, M. van de Panne, and E. Fiume, "Interactive control for physically-based animation," Proc. 27th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '00, pp. 201–208, 2000.

[4] I. D. Horswill, "Lightweight Procedural Animation with Believable Physical Interactions," Comput. Intell. AI Games, IEEE Trans., vol. 1(1), pp. 34–49, 2009.

[5] Y. Lee, S. Kim, and J. Lee, "Data-Driven Biped Control," ACM Trans. Graph., vol. 29, no. 4, p. 129, 2010.

[6] K. Yin, S. Coros, P. Beaudoin, and M. van de Panne, "Continuation methods for adapting simulated skills," ACM Trans. Graph., vol. 27, no. 3, p. 1, Aug. 2008.

[7] S. Wang, "Simple Quadruped Simulation," pp. 1–5, 2010.

[8] S. Coros, P. Beaudoin, and M. van de Panne, "Generalized Biped Walking Control," ACM Transctions Graph., vol. 29, no. 4, p. Article 130, 2010.

[9] T. Geijtenbeek and N. Pronost, "Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review," Comput. Graph. Forum, vol. 31, no. 8, pp. 2492–2515, Dec. 2012.

[10] J. Zajac, "Biped Animation Using Mathematical Expressions in Maya," Proc. CESCG'03 (Central Eur. Semin. Comput. Graph., pp. 1–5, 2003.

[11] Z. Bhatti, "Procedural Model of Horse Simulation," in 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI '13)., 2013, pp. 139–146.

[12] Z. Bhatti, A. Shah, A. Waqas, and M. Karbasi, "Automated animation of quadrupeds using procedural programming technique," Asian Journal of Scientific Research. vol. 8, no. 2, pp. 165–181, 2015.

[13] Z. Bhatti, A. Shah, M. Karabasi, and W. Mahesar, "Expression driven Trignometric based Procedural Animation of Quadrupeds," in International Conference on Informatics and Creative Multimedia 2013 (ICICM'13), 2013, pp. 1–6.

[14] J. Morimoto, G. Endo, S.-H. Hyon, and G. Cheng, "A simple approach to diverse humanoid locomotion," 2007 7th IEEE-RAS Int. Conf. Humanoid Robot., pp. 596–602, Nov. 2007.

[15] J. Morimoto, S. Hyon, G. Cheng, D. Bentivegna, and C. G. Atkeson, "Modulation of simple sinusoidal patterns by a coupled oscillator model for biped walking," in Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., 2006, no. May, pp. 1579–1584.

[16] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," Rob. Auton. Syst., vol. 47, no. 2–3, pp. 79–91, Jun. 2004.

[17] Z. Bhatti and A. Shah, "Widget based automated rigging of bipedal character with custom manipulators," Proc. 11th ACM SIGGRAPH Int. Conf. Virtual-Reality Contin. its Appl. Ind. - VRCAI '12, p. 337, 2012.

[18] Bhatti, Z., Shah, A., Shahidi, F., & Karbasi, M. (2014). Forward and inverse kinematics seamless matching using Jacobian. Sindh University Research Journal (Science Series) Volume 45 (2) . arXiv preprint arXiv:1401.1488. pp:387-392.

 ISSN – 1999-4974